Calhoun: The NPS Institutional Archive

DSpace Repository

Theses and Dissertations 1. Thesis and Dissertation Collection, all items

1990-03

A prototype production rule program for a decision support system

Buscemi, Thomas, Jr.; Masica, John Michael

Monterey, California; Naval Postgraduate School

http://hdl.handle.net/10945/18845

This publication is a work of the U.S. Government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States.

Downloaded from NPS Archive: Calhoun

DUDLEY KNOX LIBRARY

Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

http://www.nps.edu/library

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

A PROTOTYPE PRODUCTION RULE PROGRAM FOR A DECISION SUPPORT SYSTEM

by

Thomas Buscemi, Jr.

and

John Michael Masica

June 1979

Thesis Advisor: R. J. Roland

Approved for public release; distribution unlimited.

T189164

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Prototype Production Rule Program for a Decision Support System		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; June 1979
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Thomas Buscemi, Jr. John Michael Masica		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE June 1979
		13. NUMBER OF PAGES 145
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Postgraduate School Monterey, California 93940		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

EMYCIN
Decision Support System
Predicate Calculus
Production rules
Knowledge base

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A prototype production rule program, DECAIDS, for the support of decisions on computer resources was constructed using the Stanford University EMYCIN production rule system. The DECAIDS program demonstrates the use of a production rule system to support a relatively unstructured management problem. A discussion of knowledge based systems, predicate calculus and production rules is included. A tutorial section discusses

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
(Page 1) S/N 0102-014-6601 |

the user information needed for development of a backward-chaining, goal-seeking knowledge base system.

Approved for public release; distribution unlimited.

A Prototype Production Rule Program for a Decision Support System

by

Thomas Buscemi, Jr.
Major, United States Marine Corps
B.S., United States Merchant Marine Academy, 1966
M.S. University of Southern California, 1971

and

John Michael Masica
Lieutenant Commander, United States Navy
B.S., United States Naval Academy, 1969

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1979

ABSTRACT

A prototype production rule program, DECAIDS, for the support of decisions on computer resources was constructed using the Stanford University EMYCIN production rule system. The DECAIDS program demonstrates the use of a production rule system to support a relatively unstructured management problem. A discussion of knowledge based systems, predicate calculus and production rules is included. A tutorial section discusses the user information needed for development of a backward-chaining, goal-seeking knowledge base system.

TABLE OF CONTENTS

I. INTRODUCTION - 6

II. BACKGROUND - 9

III. PREDICATE CALCULUS - - - - - - - - - - - - - - - - - - 16

IV. AND/OR TREES - 20

V. PRODUCTION SYSTEMS - - - - - - - - - - - - - - - - - - 24

VI. DECAIDS KNOWLEDGE BASE - - - - - - - - - - - - - - - - 44

VII. TUTORIAL - 55

VIII. CONCLUSIONS- 80

APPENDIX A DECAIDS PRODUCTION RULES- - - - - - - - - - - 87

APPENDIX B DECAIDS PARAMETER LISTING - - - - - - - - - -104

APPENDIX C SAMPLE CONSULTATION - - - - - - - - - - - - -109

APPENDIX D ADDITIONAL EMYCIN/DECAIDS PARAMETER PROPERTIES- - - - - - - - - - - - - - - - - - - -112

APPENDIX E EMYCIN/DECAIDS PREDICATE FUNCTIONS- - - - -118

APPENDIX F DECAIDS USER PROCEDURES - - - - - - - - - - -128

APPENDIX G DECAIDS KNOWLEDGE BASE MODIFICATION PROCEDURES- - - - - - - - - - - - - - - - - - - -133

BIBLIOGRAPHY- -143

INITIAL DISTRIBUTION LIST - - - - - - - - - - - - - - - - -144

I. INTRODUCTION

Production rule systems provide a means for encoding "expert" knowledge about some domain of information and offer techniques for using this knowledge to provide answers for questions in the domain. A prototype production rule program, DECAIDS (Decision aids), was developed using current Artificial Intelligence (AI) methodology in an effort to organize decision aiding characteristics relating to computer resource allocation alternatives. The purpose of this program is to provide recommendations during an interactive consultation session. During this session, specific information is requested from the consultation system user concerning his organization's task, technology, environment, and structure characteristics. This information is then used to invoke system production rules which are used to produce the recommendations.

While the primary subjects of this paper are the production rule system and a sample categorization of managerial decision aids, it is noteworthy to mention that several other areas of study are involved. These include inferential analysis, organizational theory, predicate calculus, and decision aid characteristics. The last section in this thesis is a tutorial containing instructions for the use of the EMYCIN inference engine and procedures for designing and implementing a database (referred to as a knowledge base).

While some AI systems do attempt to model the human thought process, this paper has adopted the EMYCIN inference engine developed from Stanford University's MYCIN research program. This program, EMYCIN, permits problems to be solved using goal-directed, backward-chaining production rules. Backward-chaining is a tree traversal method. It is accomplished by searching through a tree from a goal, bottom leaf, state toward some initial, root node, state. The concept of "spaces" may also be used here to further explain that the shape of the state space determines whether forward or backward-chaining is utilized; fanning in from some goal state to an initial state represents backward-chaining [Winston, 1977].

The important thing in AI is that a software program produces "behavior" similar to that which a human can produce, understand, and recognize. Generally, the problems posed to AI are not those for which a specific algorithm can be written. For instance, a manager probably could not explain, algorithmically, how he arrived at a particular decision. He uses a myriad of facts, procedures, and experiences to tell him what to do during certain circumstances and these factors are used to produce a decision. EMYCIN was developed with the assumption that a human would use the same data that the program used but not in the same manner. The program provides a method to weigh conflicting information, calculating how much information is sufficient to achieve a recommendation, and identifying those cases where insufficient

information is available to arrive at an acceptable decision. This is what a human does when making a decision. The AI program must therefore parallel these human processes as closely as possible in order to produce an acceptable result [Scott, 1979].

The EMYCIN program is written in INTERLISP. This computer language provides an excellent basis for an AI system since the information in the knowledge base is grouped into lists, at the tree nodes, and these lists are appropriately manipulated by INTERLISP functions. This thesis provides the background materials and concepts required to design, implement, and operate an Artificial Intelligence knowledge based system.

II. BACKGROUND

The first domain in which computers have successfully demonstrated expert performance is mathematical problem solving. The speed with which computers can solve arithmetic problems is well known. The ability of the user to adequately prepare the program for a particular problem is the major limiting factor in computer performance. Production rule systems will be discussed as a method of preparation, or representation, of data used in Artificial Intelligence programs.

Artificial Intelligence (AI) is a relatively unexplored area of computer science. Simply stated, AI is the study of ideas which enable a computer to perform in a manner that resembles intelligent behavior. The ability to acquire and apply knowledge, to manipulate and communicate ideas, and the ability to reason are all parts of this intelligent behavior. A concise definition of intelligence is difficult to state because of the wide range of information-processing and information-representation concepts involved. The primary goals of AI are to make computers more useful and to comprehend the principles which make intelligence possible [Winston, 1977, p. 1].

A. ARTIFICIAL INTELLIGENCE SYSTEMS

The field of Artificial Intelligence is concerned with producing systems which operate in five basic areas. These

represent basic human activities and therefore the supporting systems are referred to as Artificial Intelligence [Nilson, 1978, p. 1-13]. The first of these areas is intelligent information retrieval.

Information retrieval, from database systems, is not a new problem to computer science. However, when the associated system is designed to produce inferences or deductions dependent upon the information in the database, the program takes on apparent intelligence. Use of natural language (i.e., English) methods to produce deductions and storage of large amounts of generally known information are the primary design considerations facing the system designer.

Expert information, in the form of rules, inference statements, is interpreted to produce values for parameters, which are the program's ultimate output. The domain of computer resource allocation based upon organizational structure is an example of an intelligent information retrieval system contained in this paper. Theorem proving, disease diagnosis, mechanical controls (robots), and automated programming are some other applications for AI [Nilsson, 1971].

One of the problems encountered in AI is the requirement for categorizing into a manageable form vast amounts of data. A system may be provided with sensory equipment which collects input data. However, unless there is sufficient information for the system, for example a robot, to test against, the interpretation of the input data may be incomplete.

The AI program, unlike conventional computer programs which are highly formatted, will possess a large knowledge base of facts from which the attainment of goals is sought. The amount of information required to accomplish a given task is an area of major emphasis to AI researchers. For example, the difficulty in programming robots to process sufficient information and to perform even simple jobs lies in the total number of possible combinations of situations which may be encountered and therefore required to be stored in the knowledge base. Accordingly, the system must have a method for selecting from among the knowledge base facts to make appropriate inferences.

This paper will explain a prototype decision support program, DECAIDS, and explain the necessary background and user information needed to use or expand the program.

B. ARTIFICIAL INTELLIGENCE LANGUAGES

As the study of AI has expanded, several programming problems have led to the search for new and more powerful methods for representing knowledge. Paramount among these difficulties has been the inability of existing languages to deal adequately with list processing and symbol manipulation. This fact has led to the development of several new programming languages, such as COMIT, IPL, LISP, INTERLISP, and SAIL to name just a few [Bobrow and Rafael, 1974].

This paper deals with INTERLISP as used by the EMYCIN/DECAIDS program. The EMYCIN system was used to provide the inference engine (inference manipulation) for the prototype managerial decision aiding program DECAIDS. INTERLISP is a relatively easy language to learn and use because it is constructed of a simple syntax. It requires no previous knowledge of high-level languages such as FORTRAN or PL/1. There are about forty common functions in INTERLISP. Approximately one-half of these are highly mnemonic arithmetic operations. The remaining functions perform the other operations required for successful list processing and symbol manipulation [Winston, 1977].

The basic structure of INTERLISP is the symbolic-expression (S-expression) which is called a list of elements. These elements may be numbers or function names. This format of s-expressions can be readily adapted to the n-tuple concept described in the predicate calculus section of this paper. The following example has three elements. (The parentheses are required of all references to INTERLISP.) [Teitelman, 1974, p. 53].

```
(PLUS 3.14 2.71)
```

In this example, PLUS is the addition function standing before the two arguments, 3.14 and 2.71, which are to be acted upon. This example also demonstrates the prefix notation which is used in INTERLISP (i.e., the function always precedes the arguments).

C. KNOWLEDGE BASE

The knowledge base, for an Artificial Intelligence program, is the data base supplied by an "expert" and operated on by a production system. This database is composed of an ordered string or strings of replacement rules. Designing and implementing a knowledge base requires the answers to some general problem-solving questions:

> What kinds of data are required? (specific facts or ideas); how should the knowledge be represented? [Should the system query the user or vice versa? (EMYCIN takes the first direction toward deriving inferences.)]
> How much knowledge is required to cover the subject? (Specific, scientific subjects lend themselves far more readily to quantification than do more subjective domains.) And what is the required information?

Finally, a knowledge base must be modeled and the tree of subject-entities (contexts) arranged so that the questions asked about the domain are contextually sensible, that is, have some direction. It serves little purpose for the program to ask questions that have no direction. The knowledge base's context tree must provide this understanding to the consultation-recommendation session. The questions asked during the consultation must be asked in a logical order to fill in the knowledge base. This can be partially accomplished by arranging the queries in an order that makes

the session flow in a smooth manner. When the entire context tree has been traversed, inferences are produced via the system's production rules.

The EMYCIN system developed at Stanford University provides a framework for building consultation programs in any quantifiable field. The domain independent components of production rule systems and backward-chaining mechanisms manipulate the information in the knowledge base. More specifically, EMYCIN/DECAIDS uses an evolving knowledge base composed of declared parameters and rules for concluding goals.

The knowledge base contained in DECAIDS is designed to support a prototype managerial decision aiding tool. While the recommendations rendered by DECAIDS are straightforward, it must be remembered that the primary goal of this research was to demonstrate a capability of designing and implementing a decision support system based on AI. While previous AI research programs have been directed toward more structured applications, the current research investigates an area which is relatively unstructured and very subjective. Previous applications, for example, the subject of blood chemistry, will produce many specific statements and rules relative to the chemical conditions affecting a person's health.

The system designer must know, and model, exactly what goal his system is to achieve before he attempts to write production rules. This requirement must be kept in mind

so that the knowledge base will be proper~~t~~ly supported by the elements of the knowledge base. The knowledge base contains two types of elements. These elements are rules and parameters which are used by the EMYCIN inference engine to support the search for a recommendation.

Each of these elements is described by a list of its respective properties. The definition and instructions concerning these properties are contained in Section VII. The rules are the sentences, (If condition...then action) statements, which imply the value(s) of parameters. The parameters are the nouns used in the sentences. One or more of these parameters will be identified as the root parameter(s) in the root context. The remainder of the parameters are used to help define, find a value for, this root parameter(s). The rules are the statements which ask for the needed values and produce the recommendations. These questions may be asked either explicitly from the system user or implicitly from the system itself.

The specific syntax for the rules and parameters are explained and demonstrated in following sections of this paper. The current knowledge base consists of forty-one rules and twenty-three parameters.

III. PREDICATE CALCULUS

The predicate calculus is a system of logic in which it is possible to express complex logical statements as well as mathematical and natural language statements. The system has rules of inference that permit valid logical deductions of new statements to be made from a set of given ones. The most often used rule of inference is that of _modus ponens_. The inference rules produce well-formed-formulas from given ones. Predicate calculus' generality and logical power are important vehicles for performing deduction [Nilsson, 1978].

AI production systems are based on the formulas of the predicate calculus. A production is a rule consisting of a situation-recognition element and an action element. Thus a production is a situation-action pair in which the left, recognition, element is a list of conditions to ascertain or test and the right, action, element consists of a list of things to do or conclude. A list may contain only a single element. When productions are used in deductive systems, the situations that trigger a production, or rule, are specified combinations of facts. The actions are restricted to being assertions of new facts deduced directly from the triggering combinations. The productions may be called premise-conclusion pairs rather than situation-action pairs. The action of triggering premises and conclusions is based upon the use of predicates and predicate logic.

Production rules are spoken of as "firing" which refers to the action taken by predicate functions [Waterman, 1976].

A programming language, using the predicate calculus logic, and specified by some syntax, is used to make assertions about some domain of interest -- to provide state descriptions about which some conclusions will be made [Winston, 1977, p. 257]. The class of expressions referred to as well formed-formulas (WFF) is the basis for the assertion clauses of a particular language. The WFF's are used as the contents of a knowledge base and are permitted values of true or false. Techniques for manipulating WFF's permit an AI program to reason about a domain and ultimately make or reach a conclusion [Nilsson, 1979]. The method of operation is that WFF's are applied, modify a knowledge base and eventually meet some termination conditions.

The well-formed-formula (WFF) is given meaning by interpreting it (the WFF) as concluding some statement about a domain of discourse. For example, the domain of interest in this research is the set of statements concerning a decision support system. Conclusions drawn from this domain involve relationships among statements in the set of WFF's.

The WFF's have values of true or false derived from the use of predicates (words or functions which direct some action be taken) whose values in turn may be true or false. The predicates perform the action of mapping elements of the domain (elements in the knowledge base) onto other elements of the domain (actually a local consultation or

session database). The WFF's are driven by the predicates of a language, the elements in the domain, and the relationships between the elements.

Each WFF can be assigned a value of true or false and these values are used to obtain a final concluding value. The values for a WFF are referred to as certainty factors. Certainty factors are based on probabilistic reasoning and represent a rule weight assigned by a system designer or by an expert from whom knowledge base information has been obtained. The WFF (premise) which evaluates successfully provides a value between -1 and 1. Those rules with true premises have their actions evaluated and a conclusion is made with a certainty equal to the premise value times the certainty factor [Davis, 1977].

A. LOGICALLY FOLLOWS AS A PROOF

The concept which allows a recommendation or conclusion to be produced is the idea of a WFF being a logical consequence of a given set of WFF's. This is formally stated as the theorem of modus ponens which is: If a statement, A, is valid and A implies a statement, B, then B is valid. The premises of the production rules are the WFF's of the domain of discourse. These premises or WFF's may further be divided into clauses representing multiple conditions with each clause being evaluated for its true value [Nilsson, 1978].

The implication of the preceding is that a "goal" in the EMYCIN system is the determination of the value of some parameter. This parameter's value is derived as a result of the inferences of the WFF's. Each rule may have one or more clauses in its premise with the clauses of one rule joined by an "AND" function. All conditions in a single premise must be true in order to fire the right-hand-side of a rule. The list of rules is a set of conditions joined by a logical "OR" function. As such, any or all of the rules may succeed and give the subject parameter a value. Which value to present to the consultation system user is a combination of certainty factors, probabilistic reasoning, and expert judgmental knowledge [Scott, 1979].

IV. AND/OR TREES

The concept of a tree-structured logic diagram is utilized in Artificial Intelligence applications to depict a graph of nodes representing state descriptions which are parameter values. A tree is referred to as a context (or a context tree depending upon the size of the domain of discourse) and state descriptions are the parameters of the context. Parameters in turn may have sub-goals used to find out or trace their values. These sub-goals are additional parameters used in other rules. Values are determined by traversing the tree and applying rules.

The tree traversal method used in DECAIDS is called "backward chaining" and is described as beginning a search at some goal state and proceeding to some initial state. To be more specific, the hierarchial structure of tree nodes, or state descriptions, where a node may have more than one parent, is properly called a graph. This may be the case with multiple rules applying to a specific situation. In the case of multiple rules, a control strategy must be implemented to select and execute rules in a logical manner. This control strategy must have some system for selecting relevant rules -- some special knowledge of the problem to be solved or how the program works. The control strategy used in EMYCIN and DECAIDS is the listing of sets of rules into groups pertinent to only certain states or declared

parameters. This grouping serves to focus the tree traversal towards the desired initial state from some goal state.

The backward-chaining traversal method is used with AND/OR trees. The name "AND/OR" is derived from the fact that the clauses in the action part of any rule are considered to be operated upon by an "and-ing" function. Separate rules in a knowledge base are considered to be operated upon by an "or-ing" function. The junctions of the rules clauses and the rules make up the nodes of the AND/OR tree.

AND/OR trees facilitate control strategies in decomposable production systems such as DECAIDS. An explanation of a decomposable system follows: A rule application only affects that component of the global knowledge (accessible to all rules) used to state the rule's premise. The decomposition of the knowledge base is represented by an atom, one parameter, at a time being affected (modified, added, or deleted). A primary benefit of the decomposable system is that redundant paths are not searched.

An example of the decomposition of a production system is a rewriting rule such as B implies Cj which produces a string of all Cj's from some arbitrary string of capital letters. The objective is to establish the sequence of rewriting rules which produces the string of all Cj's. Each step in the sequence is a decomposed part of the system. The premise clause of a system's rules form the AND nodes and junctions of multiple rules form the OR nodes in a tree.

Those rules whose preconditions, or premise clauses, evaluate to true provide the path to a desired state [Nilsson, 1978, p. 1-52]. This structure of the AND/OR tree is used in DECAIDS to select relevant rules and to calculate the final strength of each rule. These strengths or weights are referred to as certainty factors.

A. CERTAINTY FACTORS

Certainty factors provide a methodology for quantitatively supporting the reasoning of production rules. A numerical value is assigned to each rule conclusion when that rule is added to the knowledge base. This is done by the rule writer. The value assigned is not considered to be a probability but more of some "expert's" judgmental reasoning, and values are permitted to range from minus one to plus one. The certainty factors are passed along an AND/OR tree in the following manner [Davis, 1977, p. 22].

The "and" function is a minimization function affecting production rules which contain one or multiple preconditions or clauses. Minimization is effected by the fact that the conclusion of a single production rule can never be stronger than the weakest piece of information.

The "or" function is a maximization function. Accordingly, the certainty factors of multiple rules reinforce (or detract from) one another. From any "or" function the cumulative certainty factor is the algebraic sum associated with rules leading to that node. The final conclusion's

certainty factor is again the algebraic sum of the rule certainty factors leading to the root node. If a certainty factor falls below .2, an arbitrary threshold for EMYCIN/DECAIDS, the conclusion is not utilized and the situation is considered as having no rule to conclude about it [Winston, 1977, p. 245].

V. PRODUCTION SYSTEMS

As with conventional computer programs, Artificial Intelligence programs are characterized by data, operations, and a control strategy. An AI program is more specifically seen to have a global database, a production system of rules to accomplish operations, and a control strategy to determine which rules to apply and in what order [Nilsson, 1978].

A production rule program contains a set of rules. This program (DECAIDS) is the set of rules which is defined by some production system architecture (EMYCIN) referred to as a production system. Each of these rules has one or more preconditions to be verified. Upon verification of these preconditions (or premise clauses), the rule's conclusion statement will be executed. Information in a knowledge base may be modified, deleted, or added by the action of a rule [Waterman, 1976, p. 3].

In affecting parameter values, rules are considered to be modularized pieces of coding and distinct pieces of information. Each is a separate "chunk" of knowledge used in the program [Davis, 1977, p. 7]. In performing operations on a knowledge base, rules are controlled by a rule monitor and a rule interpreter. A rule monitor is a subroutine designed to affect the desired control strategy while a rule interpreter is a subroutine called by the monitor to execute rules and, thereby, determine the values of parameters.

These functions will be explained in detail later in this section.

When individual rules modify the knowledge base, no extensive changes to program code are necessary because each rule is modularized. However, it must be noted that while the rules and parameters may be added or deleted without requiring any changes in the knowledge base, their additions or deletions may affect the logic required to present a complete path of question-asking reasoning to a root node. Due care must be exercised not to disturb the backward-chaining path used to reach a logical conclusion.

In using a production system to address a problem, three areas must be addressed: problem states, the rules, and termination conditions. Problem states are the total number of alternative solutions possible to achieve a goal. This may also be referred to as the problem space. These alternatives must be formulated into some standard computer programming data structure for program use. Lists have been used by Artificial Intelligence programmers as the most appropriate data structure. Accordingly, LISP and INTERLISP are currently the most often used languages for AI applications. State descriptions serve the purpose of describing rule preconditions. The approach to a problem in an AI program is through a sequence of state descriptions and rule applications which modify a knowledge base to arrive at some termination condition. The rule monitor is responsible for

recognizing the termination state or condition as specified by the system designer.

The control strategy has the responsibilities of selecting rules, accounting for problem states (parameter values), and accounting for rule usage. There are two basic control strategies: irrevocable and tentative. The irrevocable method applies a rule with no reconsideration of its effect on a knowledge base. An example of a tentative system is the backward-chaining used in DECAIDS. Reconsideration of a rule's effect is seen as the continual computation of certainty factors along a traversal of the AND/OR tree. Control strategies may further be explained by describing two types of production systems, condition-driven and action-driven [Nilsson, 1978, p. 1-22].

A. TYPES OF PRODUCTION SYSTEMS

In general there are two types of production systems: condition-driven and action-driven. The method of interaction with the knowledge base is the deciding factor between the two. In a condition-driven system, the conditions of the premises are compared to the database and the rules whose conditions match the database are chosen to have their Right-Hand-Side (RHS) executed. The action-driven system interacts with the knowledge base by first checking the RHS's. This procedure parallels a logical implication with its "1 and 2 imply 3" statements. The system attempts to prove that 3 is true by checking for 3 in the database and, if this is

false, then proving that 1 and 2 are true -- therefore 3 is true -- and adding 3 to the database [Waterman, 1976, p. 3].

A conflict set, in a condition-driven system, is the collection of all rules whose Left-Hand-Sides (LHS) have proven true. Selection of the appropriate rule to execute is an action called conflict resolution. The most often used technique is rule ordering where each rule is previously assigned some priority value and the rule with the highest value is executed.

EMYCIN and DECAIDS are primarily action-driven systems. A premise is presented to the system to be evaluated either true or false. The premise may be proved true by the user providing an answer to a question or through deductive inference. The method for proving the premise is to examine the actions of rules to find one which would make the premise true. All clauses of the premise must prove true. The following is an example of an action-driven problem solution:

DATABASE: A F

RULES: 1. A & B & C = D
2. D & F = G
3. A & J = G
4. B = C
5. F = B
6. L = J
7. G = H

The goal is to prove H true. The system first checks the database to find H. If this fails, which it does here, the system tries to deduct that H is true by using the rules that contain H on the RHS's. The first relevant rule is 7. G is next sought in the database since if G is in the database then H is true. Therefore, rules containing G must now be tried. Rules 2 and 3 apply and these may be assumed to be rule ordered. D and F must be proven and this is accomplished via proving A true in the database, B and C are true from 5 and 4, and, finally, D and F are true, G is true and H is true. As the rules are executed, the newly proved elements are added to the database [Waterman, 1976, p. 6].

This simple example of an action-driven system should be kept in mind when the reader begins to write rules, complete the knowledge base, about a specific domain of information. More specifically, the source of a value for a parameter should be considered -- from the user or from a knowledge base search.

B. THE EMYCIN INFERENCE ENGINE

Researchers in Artificial Intelligence have recently attempted the solution of some real-world problems. Infectious blood diseases, pulmonary functions, aircraft wing structural analysis, and decision aiding systems are a few of the subjects considered relevant for AI applications. In each of these areas, large amounts of task-specific knowledge

must be collected to build a knowledge base. This information is only available from "experts" making assertions about a particular subject or through research. However, an Artificial Intelligence program must provide the desired utility before it will draw an interest. It must also be provided with a method of maintaining an evolving body of information. The use of production rules aid these two AI design criteria [Davis, 1977].

This thesis includes an example of an interactive decision aiding program (DECAIDS), designed to assist the middle/top-level manager or commander in making decisions concerning acquisition and use of automation within his organization, and when given certain organizational characteristics, to determine an appropriate organizational structure. Additionally, the method of data acquisition which provides for a system creditability and maintains an evolving knowledge base will be explained. The "Essential" MYCIN (EMYCIN) inference, production rule, engine is the programming vehicle used to accomplish deductions and produce conclusions in DECAIDS. (The name MYCIN was given to the production rule program which was first concerned with infectious blood diseases because many medicines ended with the suffix "-mycin." EMYCIN is an extension of the original system to other domains. The term "inference engine" refers to the concept of EMYCIN being a soft machine which produces inferences.) [Scott, 1979]. The inference engine modules are depicted in Figure V-1.

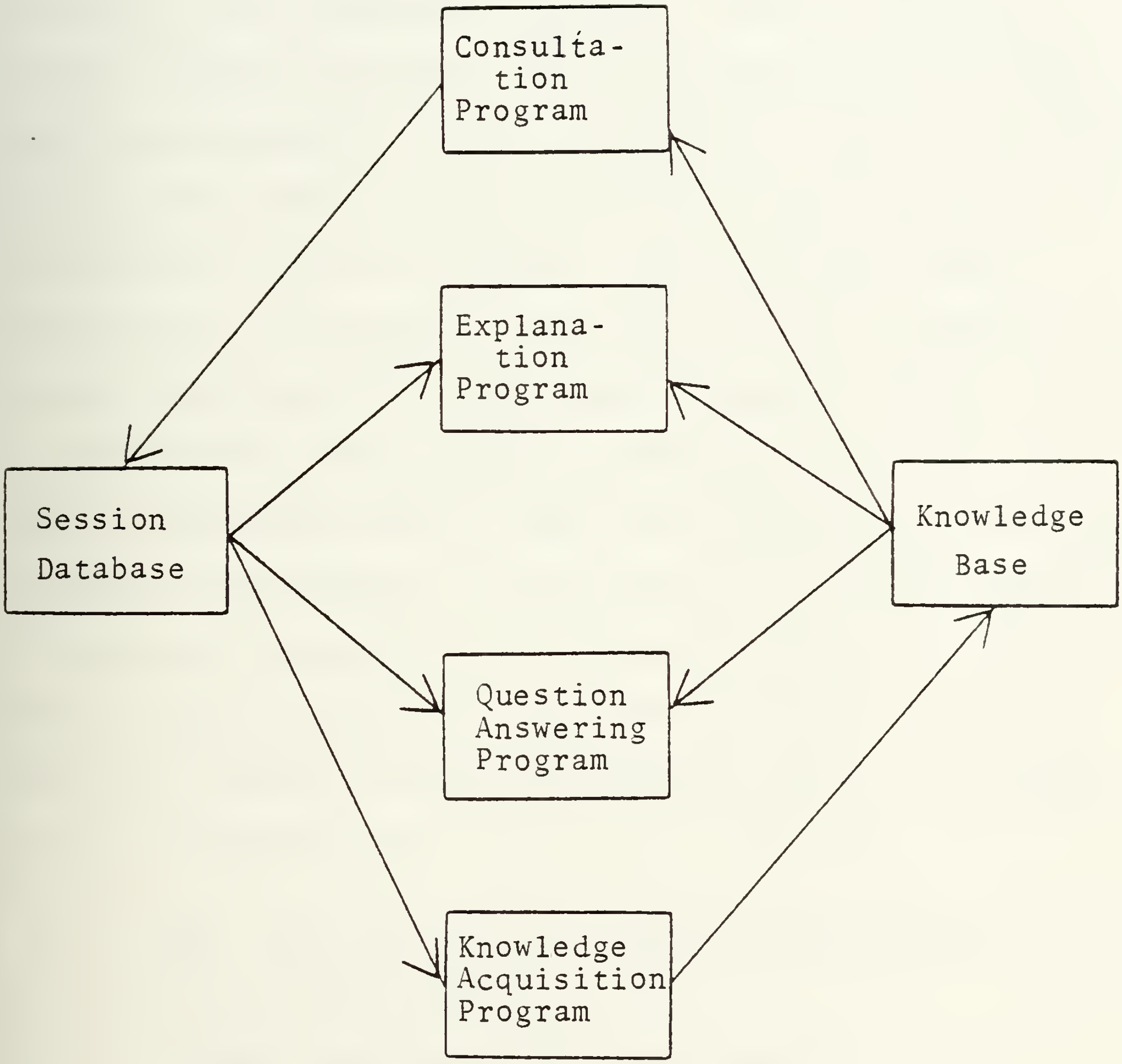


Figure V-1

Inference Engine Modules

The advice sought from the system is provided by the interactive consultation session. Three other system utility programs exist for extending the knowledge base (the acquisition program), for providing reasons for a conclusion (the explanation program), and for answering natural language questions about the knowledge base (the question-answering program). The question-answering program requires an extensive library of domain definitions and is currently not in use in EMYCIN. These modules occupy a total of 130K of nonshared code, are written in INTERLISP, and run fast enough for real-time interaction [Davis, 1977].

The DECAIDS domain specific knowledge consists of less than 50 production rules. Each rule contains, as mentioned, a premise and an action. The premise is a Boolean combination of predicate functions on associztive triples with each premise clause containing a predicate function, an object (context), and an attribute (parameter) value. An example of a clause in English is:

> "If: The structure of organization is line,"
> and in INTERLISP syntax is,
> "($AND (SAME CNTXT STRUCTURE LINE))"

The "then" part of the rule is the conclusion statement. Appendix E contains a listing of the standard predicate functions used in premise and conclusion statements. The premises are evaluated in INTERLISP to test for their

validity and the conclusion action performed if "true" is the premise value. Known conditions are saved in the "session database" by a rule adding that condition [Davis, 1977].

1. Rule Invocation Procedures

Rules are called in a backward-chaining manner which produces a depth-first search of an AND/OR tree, e.g., this search proceeds down the branches of a tree first rather than across. When given a goal to achieve, the system retrieves the list of rules whose conclusions bear on the goal -- the right-hand-sides are scanned and the goal sought for in the knowledge base by the monitor subroutine. The INTERLISP function "EVAL" is the rule interpreter for evaluating the premise of a rule. (In INTERLISP, an "s-expression" is an atom or may be a list; "s" is an abbreviation for "symbolic." An EVAL function in INTERLISP returns the value, TRUE or NIL, of an evaluated s-expression.) [Winston, 1977, p. 266]. No additional work is required. Rules whose premises are evaluated false are bypassed and those for which an answer cannot be determined have sub-goals created for them and a rule search recurs.

These sub-goals take the form of "find the value of the parameter" of the premise. "By setting up the generalized goal of collecting all evidence about a parameter, the program exhausts each subject as it is encountered and tends to group together all questions about a topic. This presents a more

orderly approach to the problem." [Davis, 1977, p. 9]. If after trying all relevant rules about a sub-goal, the certainty factor associated is still between -.2 and .2, then the sub-goal is still considered as unknown. "This may happen if no rule were applicable, the applicable rules were too weak, the effects of several rules were offsetting, or if there were no rules for the sub-goal at all." [Davis, 1977, p. 10]. EMYCIN provides the pattern matching capabilities and INTERLISP provides the evaluating functions -- the user or system writer, who completes the knowledge base, must furnish all concluding rules. If no value can be deduced, then the user may be asked for a value. A value may be an alphabetic value. A "?" will present a list of acceptable values to the user during a consultation.

In some cases, the user is assumed to always have the answer to certain questions for which there should be no need to spend time searching. Name, rank and social security number are examples of such information. In this case, where the user is immediately expected to provide a parameter value, the answer to a question, that parameter is declared with a property called LABDATA. This LABDATA property will have a value of true declared and the user will be prompted with a plain language text question requesting the subject parameter's value. This plain language text question is maintained in a parameter property called PROMPT

and the acceptable or permitted values for a parameter are kept in a property called EXPECT.

In the interest of efficiency, rules are "previewed" for clauses whose values are already known and would thus make the entire premise false. There would then be no need to evaluate this rule. When running a consultation with the highest level fault trace turned on (FT4), a rule found false during this preview will receive the following message: "Rule #, Failed in Preview Due to Clause #." The Boolean logic for this situation is False and True = False.

The "previewing" facility is accomplished with a data structure called a template used with each predicate function. Each template has the basic form of (function-context-parameter value) and uses internal flags to determine if a parameter has been previously used with this function and whether it is true or false in this clause.

The last action seen during rule invocation is the use of antecedent style or definitional rules. If a conclusion is made which matches the premise, these antecedent rules are invoked with a certainty factor of 1.0. After desired parameter values have been determined, an antecedent rule is used to produce the system's output, i.e., once the parameters describing the context ORGANIZATION are known, then an inference can be made with a definitional rule recommending the organization should use a particular type of computer installation [Davis, 1977].

C. DATA ACQUISITION

Since any domain of information can be expected to change, a capability to add and delete pieces of knowledge must be provided. The updating of an evolving knowledge base is necessary to give the subject system acceptability and recognized competence [Williams, 1978, p. 3]. The addition of knowledge to a system is accomplished by the system writer or user producing the production rules using the INTERLISP syntax described later. EMYCIN and DECAIDS scan new rules provided by the expert to find key words which indicate the appropriate predicate functions and a template, function-context-parameter-value tuple, to be retrieved. Values provided for the parameters must be a member of the parameter's EXPECT property, the list of permitted or expected values for that parameter. Upon completion of the parse of the rule, the new rule is added to the appropriate list of relevant rules of the same rule group [Davis, 1977, p. 25].

When adding new rules, direct contradictions should be avoided. While the certainty factor computations will provide a resolution, the strength of the consultation recommendation will be weakened by contradicting rules. New values and parameters must also be updated throughout the information structure of the system. While new rules may be added without regard for deleting old ones (only the true, relevant rules will be executed), parameter values

must be kept abreast of current technology. Finally, the additions of not just a single rule but the addition of an entire concept must be carefully planned when being added to a knowledge base. A single rule is easily expressed and added to the program. However, a set of rules, stated in the backward-chaining, goal-directed, manner is not a normal human method of expression. Due care must be exercised in writing a logical ordering of rules to achieve a complex concept or goal. Written first in the system writer's natural language (i.e., English) the question asked of the user (or implicitly of the system) will be the rule concluding the parameter values.

This is an example of the modeling process used to write production rules and fill a knowledge base.

1. A system designer has been tasked to accept ideas from an organizational theorist and information analyst and to produce production rules leading to a recommendation concerning computer resource utilization.

2. It has been decided that STRUCTURE, the name of an organization, may have one of two possible organizational types: LINE or MATRIX. STRUCTURE, LINE and MATRIX will be used by the system designer as parameters in the knowledge base. LINE and MATRIX are values for the parameter STRUCTURE.

3. It has further been decided that if STRUCTURE has the value of LINE, then the organization is recommended to use a large computer, graphic display, and batch processing capabilities.

4. Accordingly, the system designer decides to declare SIZE (of the computer), TYPE (of display), and MODE (of processing) as parameters with values of large or small, printer or graphic, and interactive or batch, respectively.

5. The system designer next verbalizes a premise:

If the structure is line
(and, next he verbalizes an action)
Then strongly suggest a large computer be used
(cf .8)
    strongly suggest graphic terminals be used
    (cf .8)
    strongly suggest batch processing be used
    (cf .8)

6. Finally, the rule is written in INTERLISP syntax:

```
($AND (SAME CNTXT STRUCTURE LINE))
(DO-ALL (CONCLUDES CNTXT SIZE LARGE TALLY 800))
        (CONCLUDE CNTXT TYPE GRAPHIC TALLY 800))
        (CONCLUDE CNTXT MODE BATCH TALLY 800))
```

7. The phrase "TALLY 800" is the required certainty factor syntax.

D. BASIC SEARCH METHOD OF EMYCIN/DECAIDS

Searching in EMYCIN/DECAIDS is entirely a goal-directed, backward-chaining process. Each "goal" is to determine the value of some parameter. The chaining starts with the parameter which the system builder specifies as the goal of the consultation. This is the value or values in the

GOALS property of the root context type. This GOALS property is actually a list of parameters and each one is traced in turn.

To trace or determine the value of a context parameter, the control structure accomplishes the following:

-- If the parameter's LABDATA property is "T," then the value is requested of the consultation system user. If the user responds UNKNOWN or enters a certainty factor (cf) less than 1.0, and if there exist rules that conclude the parameter, then these rules are tried. Once all rules have been tried, the system knows as much as it ever will about that parameter. If the parameter does not have a "T" value for the LABDATA property, the rules are tried first. After all have been tried, and still nothing is "known" about the value or the parameter has a PROMPT, the user will be asked the question (text) in the PROMPT property for which certain answers (values in the EXPECT property) are allowed. Here "known" means a cf greater than .2 for non-yes/no answers and a cf greater than .2 or less than -.2 for yes/no parameters. When all of this has been accomplished, the system knows as much as it ever will about that parameter.

The previously described apparatus is called the FINDOUT mechanism and is written in INTERLISP. The process of trying a set of rules is called the EMYCIN MONITOR mechanism and works as follows:

The system starts with the first rule in the list. The UP-DATED-BY property of a parameter is the list of rules

that can be used to conclude about the parameter. This UP-DATED-BY property is compiled by the GETRULES, rulewriting, program. The UP-DATED-BY property is the list that is used by the MONITOR mechanism when the MONITOR is used to find out a parameter value.

Each clause in a rule's premise is evaluated. If a clause mentions a parameter which has not yet been traced, then the FINDOUT mechanism is used to trace it. This is where the backward-chaining is demonstrated. The system now suspends what it is doing and sets up a sub-goal, i.e., to find out the parameter that is needed in the rule currently being processed. The backward-chaining drives the consultation: It is what causes all questions to be asked with the exception of the PROMPT's of the parameters in the MAINPROPS property of the contexts.

-- If the clause that is being evaluated fails, then the rule is discarded and the system MONITOR mechanism pursues the next rule in the list. If the rule succeeds, then the next clause of the premise is evaluated. When there are no more clauses to be evaluated, execution of the ACTION clause occurs.

-- If the value of the parameter is known with certainty (cf = 1) then the system seeks no further, and goes on to the next rule having executed the action. The entire process continues until each rule in the list has been tried.

The only exception to backward-chaining is in ANTECEDENT rules. These are rules that are tried as soon as all the

parameters in the PREMISE are traced as opposed to the rest of the rules which are tried in order to find out the value of the parameter in their ACTIONs. On the property list of a parameter the ANTECEDENT-IN property is a list of antecedent rules to be tried when the parameter has been traced. Once the parameters have been traced, the rule is executed -- often in an output type statement [Scott, 1979].

E. PARAMETER TRACING IN A CONTEXT TREE

EMYCIN/DECAIDS performs the following actions in order to trace the value of a parameter, called "PARM" in this example, in a context, called "CNTXT." The rules in the parameter's UP-DATED-BY property are tried one by one. The rules are listed in the UP-DATED-BY property by the GETRULES subprogram. When trying a rule, called "RULE," its SUBJECT called "GROUP," in this example, is compared to the context type of "CNTXT," called "TYPE." "GROUP" is the value of RULE's subject property and is a member of the list called ALLNAMES -- the name of the list for rule groups. "TYPE" is the value of "CNTXT's" PROPTYPE property. This value is "PROP-VAL" for all contexts.

The CONTEXT property of a rule group is the list of context types to which RULE can apply. By convention, a rule may reference parameters of the context to which it applies and parameters of descendent contexts. It may not reference parameters of contexts lower in the tree -- nearer the root. Therefore, if a rule is relevant to CNTXT,

one of two conditions may apply. The RULE may apply directly to CNTXT, and in this case, the CONTEXT property of GROUP must contain TYPE and the RULETYPES property of TYPE must contain GROUP. The other possibility is that RULE applies to a descendent of CNTXT. TYPE's OFFSPRING property is the list of context types that may be below CNTXT in the tree. The system examines each of the OFFSPRINGs of TYPE (and succeeding OFFSPRING) to try to find a descendent context type to which RULE can apply. If any such lower context is found, then the RULE is relevant. Relevant descendents of CNTXT are collected and RULE is applied to each of these contexts [Scott, 1979].

F. INTERPRETING THE PRODUCTION RULE PROGRAM

The interpretation of a production rule system is comparatively simple. The procedure involves a search through the knowledge base for a production that matches the production in the consultation data-base. DECAIDS moves back and forth searching for the relevant production and executing the rules which are true. The action of searching for a rule is made a little easier through the ease of pattern matching in LISP and INTERLISP. LISP has no organic pattern matching capabilities but can be easily programmed for this task. The basic LISP function which facilitates this interpreting is called COND. COND is the LISP branching function or conditional function. The COND function is followed by a list of premise clauses to be tested. The

pattern matching occurs at this time with the system, checking in accordance with the GETRULES "used-by" property, looking for key, or matching, expressions. More specifically, a function is DEFINE'd in LISP which examines the elements of the rules in the knowledge base and the consultation database. ("DEFINE" is a LISP function which permits a function declaration. If a match is made, a recursive call is made on the remainder of the rules to be matched. If NIL is returned, then the next clause is examined [Winston, 1977].

G. PROS AND CONS OF PRODUCTION SYSTEMS

There are several advantages and some disadvantages in using the production rule architecture. Production rules represent a form of code modularization. Each production rule is a self-contained, modular piece of code independent of other rules in execution. The advantages of modular coding are well known. Such a technique greatly aids the task of managing very large and/or evolving database. Parsing new rules to be added to the appropriate internal lists, relevant to a goal, is made easier. The production rules are a succinct form of knowledge representation. Each is easily written and retrieved. Unlike other high-level languages, no changes in previous coding need be made in order to add new rules. No other rule must be altered to ensure that the new one will be called and executed [Winston, 1977]. The use of the RULE GROUP and PROPGROUP

properties and the key word search of the function-parameter templates ensures that only relevant rules are executed [Scott, 1979].

A disadvantage in the use of production rules is the difficulty in expressing series of rules to convey a complex concept. The modularized, "if-then" coding does require some practice in the area of "backward-thinking." The rule writer must be cautious of the goal-directed invocation of rules in preparing the rules [Winston, 1977, p. 151].

VI. DECAIDS KNOWLEDGE BASE

A. BACKGROUND

The birth of decision making as a discipline can be traced to early operational analysis techniques employed during World War II [Williams, 1978, p. 4]. More recently there has been considerable debate between behavioral scientists and technically-minded analysts concerning just what decision making is. Behavioral scientists believe that decision making remains the province of organizational behavior and theory. This point of view differs drastically from that of the more technically-minded analysts who believe that this discipline is better founded in quantitative methods such as operations research and management science. The differences between these viewpoints clearly point to the fact that "decision making theory" remains relatively unformulated [Williams, 1978, p. 4].

The capabilities of automatic data processing techniques in operational and management tasks (accounting routines, record keeping, operations control, etc.) are well known. The use of these techniques by mid- and high-level managers beyond these "bookkeeping" areas is not yet significant. There are two major reasons for this lack of use by management to assist in decision making. One is the inability to adequately transfer new technologies from the research/ academic areas to the manager. However, the primary reason

for the manager's point of view may be the lack of a model which describes the organization status or situation.

B. MODEL DEVELOPMENT

The prototype program (DECAIDS), as described in this paper, represents an attempt to identify the key parameters affecting organizational managers and the decisions which they must make. More specifically, this decision support program, DECAIDS, attempts to "quantify" these parameters in order to deduce recommendations concerning appropriate computer system capabilities. These recommendations are based upon organizational characteristics described by the user.

The DECAIDS knowledge base was developed from the final technical report prepared for the Office of Naval Research by Spector, Hayes, and Crain. This report evaluates the effects of computer-based decision aids on organization structure. It was chosen as the basis for DECAIDS because it specifically addresses the subjects of decision aids and organizational structure. The knowledge base is not considered comprehensive; rather it is a skeleton which can be further expanded by subsequent research.

The current knowledge base (Figure VI-1) has ORGANIZATION as the root context. This context is defined by the organization's structure and environment. The following sections of this chapter describe the current system parameters and the ultimate recommendations which are

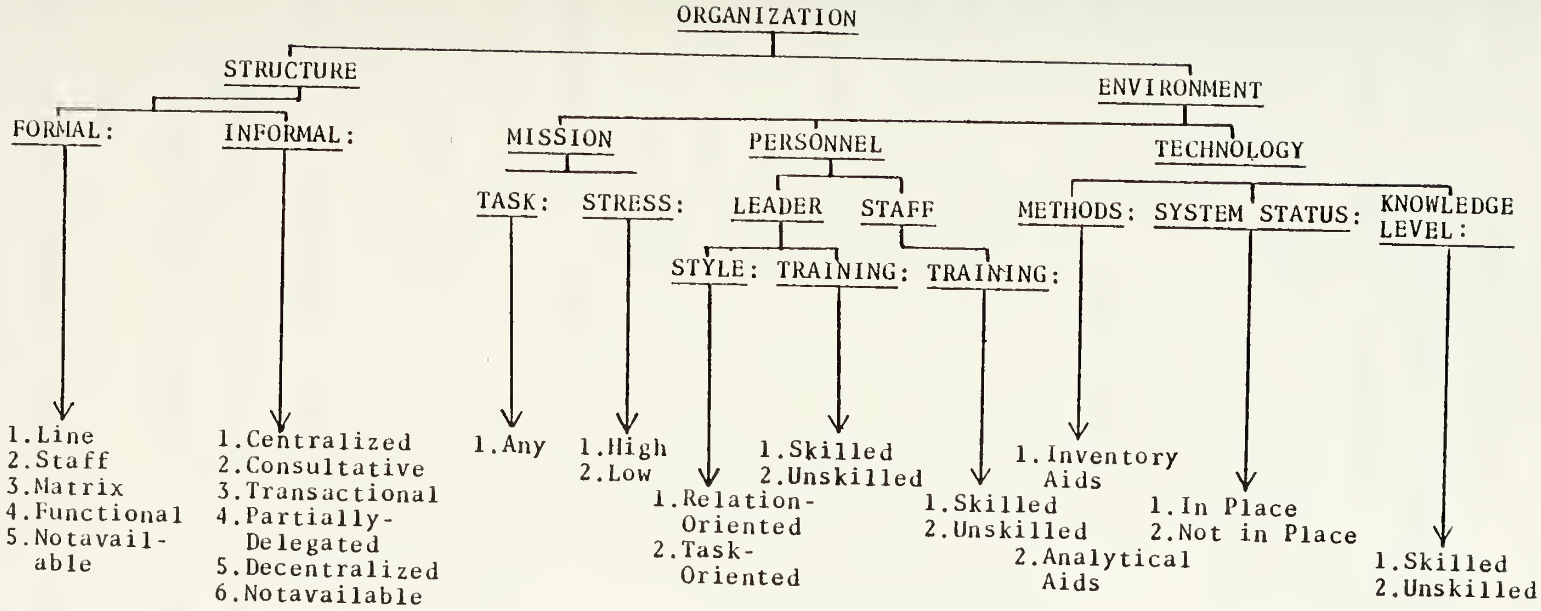


Figure VI-1

DECAIDS Knowledge Base

reached during the interactive consultative session(s). During the following discussions, DECAIDS program parameters are indicated in capital letters.

1. Formal and Informal Organization Structure

"Organization structure is concerned with the role and personal arrangements within an organization that specify authority, coordination, and communication relationships. These arrangements link functions and physical factors to manpower requirements and availability." [Spector, 1978, p. 31]. Simply stated, this organizational structure relates to the internal system of functioning groups -- the processes by which the organization operations are or should be accomplished. The organizational structure can be further subdivided into the formal and informal structures.

The formal structure (FORMAL) involves the official patterns of authority and the location of responsibility and accountability within the organization. Public and private sector usage has resulted in the identification of four basic formal structures. These four categories define the different lines of command and control, advisory, and functional relationships. The four basic formal structures used in DECAIDS are: line, line and staff (called staff), functional, and project manager (called matrix) [Spector, 1978, p. 3-2].

Line structure emphasizes direct chains of authority and unity of command while the line and staff includes an

informational and/or advisory staff to assist/guide line or operational personnel. A functional structure organizes personnel by functional activity or type of task, such as supply, planning, or communications functions [Spector, 1978, p. 3-2]. Finally the project manager (matrix) structure organizes specialists from specialized organizational units to work together with other specialists. "For example, chemical, mechanical, industrial, and electronic engineers may work together in a team with physicists, accountants, human engineers, and other professionals to develop a new product." [Tosi, 1976, p. 471]. In this structure, the individuals work to achieve the interdepartmental goals.

The informal structure (INFORMAL) describes the system of transactions, dynamic and interpersonal, which occur within an organization. These informal processes, patterns, and relationships develop quite naturally as organizational personnel interact to handle the problems and requirements of their roles in their own particular styles. While the organization's formal structure establishes the division of labor within the organization, the informal structure identifies the reality of organizational behavior and performance based upon the individuals involved.

Spector _et al_. (1978) identified five informal structures. These five informal structures which have been used in DECAIDS are: centralized, consultative, transactional, partially-delegated, and decentralized. A centralized

structure employs a focused flow of authority to a single source at the top of the hierarchy. Consultative-type informal structures maximize patterns of central control, but encourage vertical, upward communication of advice/guidance from the professional staff. Open communication, deliberation, and negotiation, not only vertically among hierarchial levels but laterally among levels is highly encouraged by transactional structures. This type structure does not preclude the ultimate decision remaining with the high levels of the hierarchy. Another form of informal structure promoting management by negotiation is the partially-delegated structure. This system distributes authority among the professional staff while increasing the need for coordination of effort. Under this arrangement, staff personnel have authority to develop action alternatives with top management retaining the right to approve, reject, or modify these options. Finally, the decentralized structure uses a high degree of delegation and dispersal of decision making authority to lower levels of the higher hierarchy.

Formal and informal structures represent the theoretical and realistic arrangement of organizations, respectively. Formal structure defines a set of decision methods and procedures designed by management in order to optimize the performance of the organization. The formal structure that is chosen reflects the past management

experience and is based upon the expected personnel configurations. The informal structure defines the actual decision methods and dynamic problem-solving processes that are used to motivate organizational performance.

In reality, these informal structures may exist in a myriad of combinations and variations. DECAIDS uses them individually and independent of one another. This program also offers the user the additional choices of declaring both formal and/or informal as not known (notavailable). If this option is used, the program contains production rules which will return suggested configurations for formal and informal structures.

2. Environment

The other parameter affecting organizations is the environment in which the organization must operate. There are three basic factors which make up this environment. They may be described as: the mission to be accomplished, the personnel required to perform the mission, and the technology available to perform the mission.

The model defines the mission in terms of problem definition (PROBDEF) and stress level (STRESS). The user is asked to define the task in terms of problem definition, either clearly defined or ambiguous. Mission stress is defined as high, low, or unknown.

Personnel environment is divided into two categories concerning the leader and staff. The system currently

contains only two parameters related to the leader. These are the leader's style (STYLE) and his level of training (LEADER-TRAINING). Leader style is defined as either relation-oriented or task-oriented. Relation-oriented refers to the leader who gives "little direction to his staff, encourages them to actively participate in setting decision-making parameters, and values the development of personnel responsibility, decentralized informal organization structures provide maximum performance." [Spector, 1978, p. 4-20]. Conversely, task-oriented leaders are defined as those who prefer far more centralization or consultative structure and are less concerned with the development of individual responsibility in the decision-making policy. Leader-training relates the level of training in the use of the computerized technical aids which the mission leader currently possesses and is defined as either skilled, unskilled, or unknown.

The staff environment is currently defined by only one parameter, staff-training (STFFTRG). This parameter relates to the staff's level of training in computerized technical aids and is either skilled, unskilled, or unknown.

The available technology relates to three parameters: computer system status (SYSSTAT), the technical knowledge level required to perform the mission, and the purpose of the system's use (METHODS). The first parameter, computer system status, refers to the implementation or operational status of the organization's computer assets. The only

alternatives currently offered are: yes, an operational system is in existence, or no, there is none in operation. The alternatives allowed for knowledge level are skilled or unskilled. The other parameter used to define technological environment deals with the use of the system. The current accepted responses are inventory aids or analytical aids. Inventory aids refer to the more administrative type uses. Analytical aids are those which concern more scientific applications.

3. DECAIDS Program Goals

The program which has been developed is currently designed to support three goal parameters. These goals are decision aid characteristics (DECAIDS), formal structure (FORSTRUC), and informal structure (UNSTRUC). Successful inference of the first parameter, DECAIDS, is the primary goal of the current program. The interactive session between the program and user is aimed at deducing the most appropriate decision aid characteristics based upon the invoked program production rules which in fact reflect the user's decision making environment.

DECAIDS is composed of four parameters that define various characteristics of computer decision aids (Figure VI-2). The current definition includes the recommended type of computer system (TYPSYS), output devices (OUTPUT), computer installation arrangement (INSTALL), and the best training/assistance alternatives for successful implementation

DECISION AID CHARACTERISTICS (DECAIDS)

TYPSYS:	OUTPUT:	INSTALL:	TRG:
1. Real-time	1. Individual-terminals	1. Pyramidal	1. Train existing staff
2. Non real-time	2. Large screen displays	2. Divisional	2. Hire specialists
3. Notavailable			3. Do not hire specialists

Figure VI-2

DECAIDS Options

(TRG). The current type systems recommended are real-time and non real-time. The output devices are either individual terminals or large screen displays. The possible installations are divisional and pyramidal. Divisional installation places authority in each division for independent systems while pyramidal installation places authority at the top of one super-system above all divisions. The final parameter reflects the training needed or assistance required by the organization in order to implement a computer based decision aiding system.

The two other goal parameters, referring to formal and informal organization structure, are invoked when the user responds that either or both of these structures are not available. When this answer is indicated, production rules for FORSTRUC and/or UNKSTRUC result in the recommendations for the use of either line and/or centralized structures are made.

VII. TUTORIAL

A. OBJECTIVES

The EMYCIN system was designed and implemented by a research group at Stanford University, Palo Alto, California. It contains the essential components needed to create and support an interactive consultation program. These essential components are the consultation program, predicate functions and their translations, the explanation subprogram, and the question-answering subprogram. (This last feature is currently not functioning in EMYCIN.) The subject domains of the program range from human blood disease diagnosis to structural analysis [Scott, 1979]. The domain of the prototype program developed in this paper is decision support system characteristics and is named "DECAIDS." In order to implement this domain, a knowledge base was created and fitted into the EMYCIN format.

The objectives of the tutorial follow:

1. Introduce the computer-naive user to the EMYCIN system and the DECAIDS program.

2. Provide users with the required background and documentation in order to further develop DECAIDS or develop a unique program of their own.

3. Demonstrate the basic features of a knowledge base system through the use of a very simple prototype system.

4. Introduce the INTERLISP programming language and provide users with sufficient information concerning its syntax and functions for use with the DECAIDS program.

5. Produce an interactive system which will provide managers with recommended decision aid characteristics based upon their individual organization structure and environment.

The first and second objectives are related and will provide users with the more basic concepts involved in producing a working system. It is assumed the user does not possess a background in computer science and every effort will be made to explain these concepts in an understandable manner.

This tutorial is intended to provide an introduction to the use of INTERLISP. The documentation contained in the tutorial and Appendices F and G is intended to furnish users with sufficient knowledge of INTERLISP so that they can work with the DECAIDS program.

Finally, the use of the prototype DECAIDS program demonstrates knowledge base features and provides the basis for the fifth objective. It is anticipated that the DECAIDS program will be greatly expanded and improved by future users.

B. KNOWLEDGE BASE SYSTEMS

The building blocks of the knowledge base are contexts, parameters, goal-parameters, and production rules. The first step in the creation of a knowledge base is the creation of

the tree structure which represents the subject domain. (Figure VI-1 provides the logical tree structure or knowledge base of the DECAIDS program.) The selection of the subject creates the root-context for the context tree. The root context will have one or more goal-parameters which represent the ultimate "recommendations" to be inferred from production rules. The various branches of the context tree are represented by parameters used to describe the subject domain of the tree. The ultimate objective is to write a set of production rules which relate parameters with appropriate goal-parameters. Goal-parameters can be defined as those parameters which are concluded from the production rules. This objective is accomplished by writing questions about the system parameters in natural language (i.e., English). The responses to these questions will constitute the information contained in the knowledge base.

The system parameters must be declared next. Section C of this chapter and Appendix D contain definition and examples of the various parameter properties. These properties must be provided by the system designer when the parameters are declared (entered on the computer system).

The knowledge base system production rules are written in the following format:

If: parameter_1 = value_1 and
parameter_2 = value_2 and
parameter_m = value_m
Then: goal-parameter_1 = goal-value_1 (cf), and
goal-parameter_n = goal-value_n (cf).

Each value for the goal-parameters will have its own certainty factor (cf) assigned. A certainty factor is a relative weight based upon probabilistic reasoning by the expert who provides the knowledge base. In EMYCIN/DECAIDS, these certainty factors range from -1.0 to 1.0 in increments of 0.1.

The successful construction of a knowledge base depends greatly on the fact that conditional statements are related to all of the declared parameters. These conditional statements are actually a set of backward-chaining rules which eventually conclude the declared goal-parameter(s). For example, one of the DECAIDS program current goal-parameters is "DECAIDS." Therefore, production rules must eventually assign "DECAIDS" some value, i.e.,

If: Parameter X is Value Y
Then: Conclude "DECAIDS" equals (recommendation).

The value of parameter X is provided to the program during the interactive session. The recommendations for this particular goal-parameter (DECAIDS) consist of the four

parameters called TYPSYS, OUTPUT, INSTALL, and TRG. These parameters are more thoroughly explained in the preceding chapter.

The successful implementation of this type system depends upon providing a chain of reasoning that will "find" parameter values which lead to the "selection" of desired goal-parameter values. This can be accomplished by first writing the conditional statements related to parameters that will conclude the goal-parameters in English. From this format these rules can easily be transposed into the INTERLISP language. The EMYCIN monitor selects the order in which conditional statements will be processed. The system designer can control the direction of the interactive session by ordering the parameters (of the context's MAINPROPS property) in a manner which makes the session flow easily. This is accomplished because the system asks its questions (PROMPTS) in the same order that these MAINPROPS are listed.

C. SYSTEM PROPERTIES

The following properties are used in the EMYCIN/DECAIDS program. They are used to implement system additions after the appropriate production rules have been written.

1. Contexts

The knowledge base is centered around the "object-attributed-value" triples. The object portion of this triple is the context. A context is some entity made up of related parameters. Each system that is constructed must have at

least one context (or root) which is the subject for that system.

In DECAIDS, the context is "ORGANIZATION." The relative simplicity of the current DECAIDS knowledge base lends itself to the use of this single context. In more complex subject domains, it may become necessary to organize system attributes into multiple contexts. For example, EMYCIN uses PERSON (the root context), CULTURES (the results of infectious cultures), and ORGANISMS (the types of organisms obtained from cultures and treated which is to be prescribed) [Davis, 1977]. System designers must decide a priori, based upon system complexity, how parameters are to be organized into contexts. It is noteworthy to mention that with proper organization, even relatively large systems can be frequently organized under one context. However, if multiple contexts are used, they must be arranged in a context tree which allows the production rules to refer to parameters of more than one related context. This is accomplished with the context properties ASSOCWITH and OFFSPRING. These properties are lists of ancestors and descendent context types, respectively.

Contexts are declared in a manner similar to that used to declare parameters. (This is true because contexts are also parameters.) These specific procedures are discussed in the following section.

The primary properties which must be declared for contexts are: MAINPROPS, TRANS, PROPTYPE, TYPE, RULETYPE,

and GOALS. Other properties, included as Appendix D, are necessary for describing systems which are more complex than DECAIDS. These names are system names and are explained as follows.

MAINPROPS - These are the main parameters which describe a context. These parameters are declared as a list and are used to "trace" or define the context. The consultation session includes an interactive phase where these MAINPROPS are asked of the user. User responses will invoke the appropriate production rules to produce the consultation recommendations. The order in which these parameters are listed in the property of MAINPROPS assists in providing a more logical or "coherent" consultation for the user.

TRANS - This property is the literal translation of the context (or parameter). This definition describes how that context will be translated in the program. In DECAIDS, the TRANS of the context ORGANIZATION is "the organization."

PROPTYPE - This property is used by the system to identify to which context a particular parameter belongs. DECAIDS contains one PROPTYPE for ORGANIZATION, PROP-ORG.

TYPE - This property is the name used to identify contexts. For example, successive consultations involving the context ORGANIZATION will be titled "ORGANIZATION 1, ORGANIZATION 2, ORGANIZATION. . .".

RULETYPE - This is a list of the rules which must be searched in order to find a particular parameter. ORGANIZATION

currently has only one ruletype, ORGRULES, which contains all rules to determine the goal-parameters. This name was chosen to stand for "organization rules."

GOALS - This is a list of goal-parameters which are applicable to the context. More than one goal is allowable. This allows for the more complex systems to be represented by single contexts. Current goal-parameters for ORGANIZATION are DECAIDS, UNKSTRUC, and FORSTRUC.

2. Parameters

Parameters are defined as attributes which describe a given context. In the knowledge base, "object-attribute-value" triple, parameters are the attributes. These attributes may be thought of as questions to be asked (of the user or the system) that describe the context. For example, the system designer will ask the following type of questions, "Is the 'parameter' of the 'context' a 'value'?" The values correspond to appropriate answers to the designer's questions. These are pre-specified (if appropriate) by declaring them in the EXPECT property of the particular parameter and are explained later in this section.

The use of parameters in production rules has been discussed. Parameters are contained in the various lists with names of the form PROP-type. This form indicates a prompt which the user will answer. These lists are further collected into either PROPGROUPS or AUXPROPGROUPS. PROPGROUPS initially contain the reserved work PROP-VAL, which

is the PROPTYPE for contexts, and eventually contain each parameter group declared in the context's PROPTYPE property. (Keep in mind that contexts are declared in the same manner as parameters.) AUXPROPGROUPS are lists of auxiliary parameters which serve varying purposes. The most useful of these purposes is defining a RULEGROUP. A RULEGROUP will be explained in the following section.

The most frequently used parameter properties in the DECAIDS system are: TRANS, PROMPT, EXPECT, REPROMPT, and LABDATA.

TRANS - This is the literal translation of the parameter. TRANS is declared in the same manner as contexts.

PROMPT - This property is the natural language text question which is asked of the user concerning each parameter. Care should be taken when composing prompts so that the consultation dialogue makes sense to the user. (The context's MAINPROPS property can be used to assist in making the consultation flow smoothly and logically. Parameter PROMPTS are asked in the same order the parameters are listed in the MAINPROPS.)

EXPECT - These are the accepted or "expected" responses to the PROMPTS. The specific values may be supplied by either the system designer or user. In order to specify that anything is an appropriate answer, the word "ANY" should be entered as the EXPECT value of that parameter. If a parameter has a PROMPT, it must also have an EXPECT value.

REPROMPT - These are additional natural language text statements which are used to further explain the question asked by parameter PROMPTs. They are of great value to the designer and user because they can remove ambiguity concerning PROMPT meanings. They automatically list the accepted responses which the system will recognize. This property is invoked when the user responds with a question mark when asked a parameter PROMPT.

LABDATA - This property is a system key that indicates that the user will provide a value for that parameter. This is done by entering "T" as the value for LABDATA.

An example of these various properties is provided for the following DECAIDS parameter:

FORMAL: (Parameter name)

TRANS: (The organization's structure)

PROMPT: (The formal structure of the organization can be defined as either line, staff, matrix, functional, or not available. If further explanation of these terms is needed, type a question mark. What is the organization's formal structure?"

EXPECT: (Line, Staff, Functional, Matrix, Not Available)

LABDATA: T

REPROMPT: (Line-emphasizes direct chains of authority and unity of command. Staff-includes an informational and advisory staff to assist and guide operational personnel. Functional-arranges personnel by functional activity such as logistics, communications, etc. Matrix-draws personnel from across departmental lines.)

3. Rulegroups

All rules must be assigned to groups called rulegroups prior to being declared (entered on the system). If no rulegroup is defined, the rules will not compile. Rulegroups are determined by the type(s) of contexts to which a rule may apply. Generally, a rule is applicable to the lowest context in the tree whose parameters appear in its PREMISE or ACTION. The group for the rule must be in the RULETYPES property of the applicable context type(s). In most cases, the RULETYPES property will be a list of a single group. All rulegroups are members of the parameter group called ALLNAMES. The use of ALLNAMES is covered in the GETPARMS section of this chapter. Before entering rules, it is necessary to define and initialize all rulegroups which are named "typeRULES." The procedure for accomplishing this is to type "SET(typeRULES NIL)" and then define the group by the GETPARMS routine. (The word type in lower case letters refers to the specific type used. In DECAIDS, the rulegroup is called ORGRULES. This stands for organization rules.)

Rulegroups have the following properties:

CONTEXT - This is the list to which rules of this type apply.

SVAL - This property tells how to translate the reserved word "CNTXT" in rules of this type.

CTRANS - This is a phase in English (a translation) describing what context types the rules apply to. This

translation fills in the blank in the EMYCIN system phrase, "this rule applies to ____." This explanation precedes actual rules when rules are actually printed by the PRINTRULES routine. The use of this routine is explained in this next section of this chapter.

D. GETTING STARTED

1. Accessing the ARPANET

The DECAIDS system currently resides on a computer at the Information Sciences Institute of the University of Southern California. Access to the system may be accomplished through the ARPANET. The different uses and further background material on the ARPANET are not contained in this paper. The specific procedures used to access the DECAIDS program on the ARPANET are contained in Appendices F and G.

2. GETPARMS

The EMYCIN program contains a routine, GETPARMS, which is used to declare all contexts, parameters, and rule groups. After logging onto the ARPANET and entering the EMYCIN executive file, EMYCIN.EXE, (as discussed in Appendix F) the procedures outlined in Appendix G are used to enter, edit, or delete parameters.

The most frequently used parameter properties used in DECAIDS are: TRANS, EXPECT, PROMPT, LABDATA, and REPROMPT, as discussed previously. The most important of these properties is PROMPT as it "prompts" the natural language question which will be asked of the user concerning

a particular parameter. The PROMPT should be written in such a manner as to present a logical dialogue to the user. The LABDATA property is a system key which indicates that the appropriate value of a parameter must be obtained from the user.

3. GETRULES

The GETRULES routine is used to declare system rules after the various parameters have been entered. These system rules are written to arrive at a final goal and thereby conclude the value of a parameter. Appendix G contains the specific procedures to be followed when using GETRULES.

Rules are entered in two parts, PREMISE and ACTION. Following the final parentheses in the ACTION clause, the rule is checked for syntactic validity and an error message is returned if an error is detected. If the subject of the rule cannot be deduced, the user is asked to confirm the rule group. The proper response is "Y" (yes) if the offered rulegroup is correct or the rule group is entered.

There is a useful feature that may be used in conjunction with either GETRULES or GETPARMS. If, during the course of entering rules or parameters, the user discovers the necessity of returning to the other routine, he may do so by typing "RULES" (in GETPARMS) or "PARAMETER" (in GETRULES). This facilitates writing a set of parameters and then calling GETRULES to declare the relevant rules.

The recommended format for entering rules is the INTERLISP syntax. A "terse" English form is available but requires much more typing. Additionally, in preparing the rules via the INTERLISP syntax, the concept of writing a set of rules to produce backward-chaining (which drives the direction of a consultation) becomes more apparent. Most of the premise clauses will be calls to the predicate functions SAME, NOTSAME, or KNOWN. These can be written as:

```
INTERLISP:                     terse meaning:
(SAME CNTXT parm value)        parm = value

        or

(NOTSAME CNTXT parm value)     parm ≠ value
```

The following example is from DECAIDS:

```
(SAME CNTXT STRESS LOW)        STRESS = LOW
```

Similarly, the numeric predicates can be entered as:

```
INTERLISP:
GREATERP*  (VAL1 CNTXT parm) number)

Terse meaning:
parm value  =  number
```

No numeric predicate functions are currently used in DECAIDS.

ACTION statements will contain functions that conclude about one or more of the context-parameter-value triples. A certainty factor (cf) for the triple is specified in the rule's ACTION. This certainty factor will be modified by the certainty of the rule's PREMISE. The function "$AND"

sets the reserved word TALLY to the certainty of the PREMISE, defined to be the minimum of the values returned by evaluating the PREMISE clauses (only SAME and THOUGHTNOT return numbers). The conclusion may be written as:

```
        (CONCLUDE CNTXT parm value TALLY cf)
        parm = value (cf)
```

The following example is from DECAIDS:

```
        (CONCLUDE CNTXT TYPSYS REAL-TIME TALLY 800)
        type system = real-time with a certainty of 0.8
```

The certainty factors are actually written in the range -1000 to 1000. This range represents the -1 to 1 mentioned earlier. The system writer should use the numbers in the preceding range in a call to CONCLUDE or other ACTION functions.

The function DO-ALL is used to conclude about the several parameters which comprise any multi-valued parameter, such as DECAIDS. Once a goal-parameter has been traced, the rule calling for PRINTCONCLUSIONS will be evaluated true and an output statement will be generated. (See rule 001 in Appendix A.)

4. Declaring the Treeroot and the RULEGROUP Type

Once it has been decided what the system objective is, the context tree is designed and filled in. It is necessary to make the following system declaration. From the EMYCIN.EXE, (not GETPARMS or GETRULES) the "SET" command is used to define the context tree root in the following format:

```
_SET(TREEROOT rootcontxt)
_SETQ(ROOTTYPE (GETP 'rootcontext'TYPE)).
```

An example of the procedure using a DECAIDS example follows:

```
_SET(TREEROOT ORGANIZATION)
_SETQ(ROOTTYPE (GETP 'ORGANIZATION'TYPE)).
```

The "_" is the EMYCIN.EXE prompt. Upper or lower case letters may be used to set the rootcontext name. This function should be performed along with "SETting" the RULEGROUP:

```
_SET(ruletype)
```

In DECAIDS:

```
_SET(ORGRULES).
```

The preceding examples contain capital letters because capitals were used throughout the DECAIDS program.

In order to save items such as the treeroot and rulegroup declarations, it is necessary to edit the CHANGESCOMS files. This file is a list delineating what should be stored on the CHANGES current program file. With the command "MF CHANGES," the editions, additions and deletions to GETPARMS and GETRULES are saved. However, the following procedure must be used to save the treeroot and rulegroup. (The "_" is the EMYCIN prompt and the "*" is the INTERLISP prompt.) The following is an example from DECAIDS:

```
_EV CHANGESSCOMS                  (This command edits the
                                  variables that follow.)

*(-1 TREEROOT)                    ("-1" means insert the
                                  following before the first
                                  element in the list.)

*SET(TREEROOT ORGANIZATION)

*SETQ(ROOTTYPE (GETP 'ORGANIZATION'TYPE))

*SET(ORGRULES)

*OK                               (This exits the editor and
                                  the file must be saved with
                                  MF CHANGES.)
```

5. Saving Changes to Rules and Parameters and Deleting Changes

Each update, addition, deletion, or edit to the system knowledge base is referred to as a CHANGE. Each CHANGE requires the complete recopying of the entire program. The command used to save these changes is "MF CHANGES." A new file is created when the user enters the system each time. This file is comprised of all previous information that has been entered plus the new entries to the knowledge base. Accordingly, there is no need to keep multiple copies of the previous CHANGES after creating a file of new CHANGES. These "old" copies may be deleted by using the command, "DEL CHANGES..(number to be deleted)." If no number is specified, the lowest number version (or oldest) will be deleted. (During the login procedures, the entire file status is presented to the user. The CHANGES file will indicate exactly what number the user currently has in use.) Occasional naming conflicts have occurred with the operating

systems, TENEX and TOPS-20, resulting in unwanted change deletions. It is therefore recommended that the user maintain at least two or three changes as "insurance" against losing everything that has been entered.

6. Loading CHANGES

In order to run a consultation, it is necessary to concatenate the knowledge base with EMYCIN.EXE. This is done by the following command, "LOADEM CHANGES." The file which is currently in use has this command included in the executive login procedures and therefore does not have to be entered by the user.

7. Displaying Parameters

The following PRINTPARMS command should be used to display all, or part of the parameters entered in the knowledge base:

```
"_PRINTPARMS(parm sort.by.group linelength file)
```

where parms may be a list of parameters, a list of parameter groups, a single parameter group, or NIL meaning all parameters. The term "sort.by.group" is T or NIL. T means that an alphabetical index is printed first, showing which group each parameter belongs in. NIL indicates that the parameters are listed in alphabetical order regardless of the group to which they belong. Linelength is the length of the line to be used, i.e., 72, 78, or 80 spaces. File is the name of the file in which to write the information. If T is used,

the parameters will be written to the terminal. The PRINTPARMS command may be used within the DRIBBLE command (which is explained in Section 9 of this chapter) to write the parameters into a separate file. An example of this command follows:

```
_PRINTPARMS(NIL T 72 T).
```

8. Displaying Rules

The procedure for displaying knowledge base rules is similar to the PRINTPARMS command. The command is:

```
_(PRINTRULES rules mode).
```

In this command, rules is a list of rules, a rule group, and mode indicates how the rules are to be printed. Mode includes these options:

B - both in English and INTERLISP

E - in English

L - in INTERLISP

J - for justification which permits inclusion of author's name.

9. Creating a File in the EMYCIN.EXE

In order to create a file of the knowledge base contents, the following sequence of commands is used:

_DRIBBLE(filename)	where the filename is of the user's choice and DRIBBLE opens a typescript for the filename.

_PRINTPARMS (or PRINTRULES)

_DRIBBLE The last DRIBBLE command closes the file.

10. Running a Consultation

Appendix F contains the specific procedures that are required to run a consultation. This section contains supplementary information that is not found in that appendix.

The following special options are available when running a consultation, "FT 1, 2, 3, 4, or carriage return "(no options)." FT stands for fault trace. The numbers that follow FT indicate the level of the "trace" desired with 1 being the lowest and 4 being the highest degree of fault, or rule, tracing. Fault trace 4 (FT4) will present each rule number as it is being sought, indicate that the FINDOUT routine is tracing the appropriate parameter, and complete with the FINISHED routine when a rule has been completely traced. Once the rule is evaluated as true, this is indicated by a message "RULE (#) SUCCEEDED."

At the other end of the scale, FT 1 will show those rules which have succeeded and then display their ACTION statements.

The user can request the following special features by entering one or more of the following options with spaces and ending with a carriage return:

I -	requests instructions to be printed.
OLD -	consider a previously-saved case (number(s) will be requested).
SAVE -	create and save file(s) for cases discussed in this consultation.
NOPR -	do not print out old questions and answers when running old cases.
SUMMARY -	summarize old session data, rather than printing out each question and answer.
UPDATE -	update old session with new information
TER -	enter terse mode.
TAB -	tabular entry mode.
TS -	write out a typescript file of consultation.
N -	(a number) reconsider previously saved case n
QA -	enter the question/answer module immediately skipping the consultation (currently turned off in EMYCIN). The terms "case" and "patient" remain from the original EMYCIN system referring to a medical consultation.

The following instructions are printed if the user responds "Y" when asked if instructions are desired:

```
    Please answer the following questions, terminating
each response with RETURN (CR).  To correct typing errors,
use the DELETE key to delete characters, (ctrl)W to
delete a word and (ctrl)L to delete the whole line.
    If you are not certain of your answer, you may modify
the response by inserting a certainty factor (a number from
1 to 10) in parentheses after your response.  Absolute
certainty (10) is assumed for every unmodified answer.  It
is likely that some of the following questions cannot be
answered with certainty.  You may change an answer to a
previous question in two ways.  If the program is waiting
for a response from you (that is, has typed "**"), enter
CHANGE followed by the number(s) of the question(s) whose
answers will be altered.  You may also change a previous
```

answer at any time (even when the program is not waiting for a response from you) by typing (ctrl)F (Fix), which will cause the program to interrupt its compilation and ask what you want to change. If the response to (ctrl)F is not immediate, try typing the RETURN key in addition.) Try to avoid going back because the process requires reconsidering the case from the beginning and therefore may be slow. Note that you may also enter UNK (for unknown). If you do not know the answer to a question, ? if you wish to see a more precise definition of the question or some examples of recognized responses, ?? if you want to see all recognized responses, the word RULE if you would like to see the decision rule which has generated the questions being asked, the word WHY if you would like to see a more detailed explanation of the question, or the letters QA if you would like to interrupt the consultation in order to ask questions regarding the decision made so far in the consultation. If you are ever puzzled about what options are available to you during a consultation, enter the word HELP and a list of options will be listed for you.

```
      Sample response (user input follows "**")
          Does the patient have a risk factor
          for tuberculosis?
          **?
          One or more of the following are considered
          risk factors for TB:  A) Positive PPD (5TU),
          B) History of close contact with a person
          having active TB, C) Household member with
          a past history of active TB, D) Chest x-ray
          showing apical scarring, E) Granulomas
          seen on biopsy of any organ tissue.
          Expected responses are:  YES  NO
          Enter HELP for user options.
          ** YES
```

```
SUMMARY:
      (Type ctrl-O to abort printout)
      UNK - answer not known
      ?   - rephrases the question and gives examples
            of recognized responses
      ??  - prints a list of all recognized responses
      RULE - prints the current decision rule
      QA -  program enters question-answer mode
      CHANGE - go back and re-request answer to question
            number # performance.  Your comments will be
            forwarded to those in charge of the MYCIN
            program.
      WHY - gives high-level explanation of the current
            reasoning chain that provoked this question.
      HOW # - explain HOW the system will achieve a goal
            referred to by number # in a previous explanation.
```

EXPLAIN - provides a more detailed explanation of a previous answer given by a WHY command.
FORGET - resets the explanation of the reasoning chain back to the lowest level, as if you never used the WHY/EXPLAIN commands.
STOP - halts the program, saving the current case on a disk file, retrievable at a later date.
HELP - prints this list. [Scott, 1979].

Once the user has answered the system queries concerning instructions and options, the consultation will begin. When a goal-parameter is found, the conclusion rule, currently called PRINTCONCLUSIONS, will be triggered and the consultation ended. An example consultation is contained in Appendix C. Fault trace 4 (FT 4) was selected for this sample consultation.

11. INTERLISP EDITOR

Chapter Nine of the INTERLISP Reference Manual fully describes the editor used with the DECAIDS program [Teitelman, 1974]. The editor is entered from the EMYCIN.EXE by typing "E" and may be reached from GETPARMS or GETRULES in order to change parameters or rules. The following is a short list of the most often used editor commands:

n - (n is a positive integer) move to the nth element of the list where the element is a parenthetical expression.

p - print the current expression, used with GETPARMS or GETRULES.

-n - move to the nth element from the end.

(-n X) - insert X after the current element.

B X - insert X before the current expression.

: - delete the current expression.

: X   - replace the current expression with X.

OK   - ends editing.

The "MF CHANGES" command is used to save all the editing which has been done.

12. Miscellaneous DECAIDS Notes

The following notes are included as information that applies to the DECAIDS program.

The INTERLISP compiler compiles upper and lower case letters differently. Either may be used to fill a knowledge base. For example, "DECAIDS" will compile differently than "decaids" and subsequent use in rules will result in an error message stating that there are not rules to conclude one or the other. Because of this problem, DECAIDS was entered on the machine in all capital letters.

The system permits a consultation user to respond with "unknown" to a request for information which is not known to the user. This "UNK" means that the certainty factor for the rule should be set to less than .2, the system's arbitrary limit for acceptable knowledge about a parameter. Therefore, if the system writer desires to provide some other certainty factor about an unknown condition, he must offer a substitute response for "UNK" in the expect values. "NOTAVAILABLE" was the choice used in DECAIDS.

Rule 035 is the print statement for the goal-parameter UNKSTRUC. If UNKSTRUC was not known, then a recommendation was made about the informal organization

structure to be recommended. If UNKSTRUC was known, no output was necessary. To "turn off" the output, even though UNKSTRUC had been traced, the HEADER in the PRINTCONCLUSIONS function (Appendix E) was left at NIL.

In rule 034, a parameter in the rule PREMISE is also used in the ACTION. In the parameter's USED-BY property (a system property not used by the designer) the note SREFMARK 34 appears. This means that the parameter is in the PREMISE and ACTION of rule 034 and is a system flag to prevent various search and circularity problems.

The parameter property REPROMPT is used to locate the text presented to the consultation user if and when he responds with a "?" for an expected parameter value. The system designer includes further explanation of expected values or the parameter definition in the REPROMPT.

The "WHY" response to a question rather than an allowed expected value produces an explanation of the current reasoning chain that provoked the current question. For the "WHY" question to work, the system writer must previously have used the LISP SET command to set the value of FINDBESTPARM. This value is used to an EMYCIN to provide text to explain the reasoning chain.

If intermediate values are desired to be known, not necessarily those of the goal-parameters, the system writer need only write a rule to "PRINTCONCLUSION" for that parameter.

VIII. CONCLUSION

It was the purpose of this thesis to demonstrate the implementation of a particular domain of information with the Artificial Intelligence system known as EMYCIN. The demonstration illustrates that even semi-structured and subjective domains can be studied using AI technology.

The domain chosen for this paper was that of managerial decision aiding. This particular subject contains many intuitive thought processes. The uncertainties and impreciseness of intuition lends itself to one of the goals of Artificial Intelligence which is the capability of producing relevant, acceptable assertions in the light of incomplete and uncertain information.

Recommendations are offered concerning possible extension of the current, single context knowledge base to a multiple context tree. Also, the motivations for further AI work are established and a summary of the research conducted and accomplishments are presented. The motivations mentioned represent the current interest for continuing Artificial Intelligence research and development.

A. EXTENDING THE DECAIDS CONTEXT TREE

The current domain of information is described by a single context and related parameters. It is intended to be a prototype domain for a decision aiding support system and an example of how to structure a knowledge base. It is

expected that the knowledge base will be extended to include a more complex context tree supporting a far more thorough treatise of the decision aiding support process. The following information is directed toward the individual who will extend the current knowledge base.

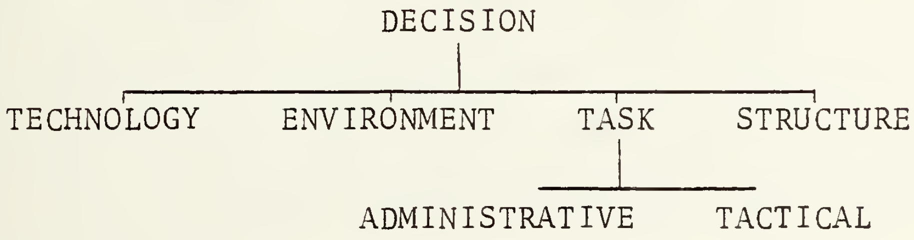
Prior to doing any coding or entering of any rules or parameters, it is strongly advised (cf 1.0) that the system goal(s) be explicitly defined. This means not only deciding that the system is to provide some form of advice but, more specifically, to write out the natural (i.e., English) language questions which will be used to trace parameter values. Entering rules and parameters into the DECAIDS knowledge base is not unlike any other high-level-language coding where a flowchart of operations and data manipulation are appropriate. A system designer is reminded that the primary emphasis for extending a knowledge base is a consistent line of questions to be asked in order to trace parameter values. Eliminating or ignoring the requirement to specify the questions to be asked can only lead to confusion.

When to define a context or multiple contexts may present the system designer with some confusion. It is recommended that the designer review the notes on backward-chaining and the concept of inferences. This information (Chapter VII) will provide the necessary background for constructing rules which terminate in the determination of values for parameters.

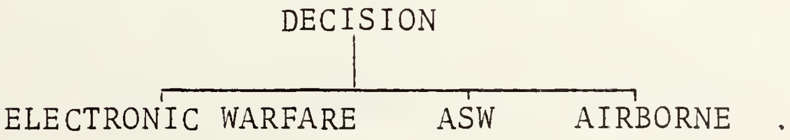
Most often a single context will suffice for a small to medium size knowledge base. However, to fully describe the management decision aiding characteristics about some subject (i.e., electronic warfare, major procurement systems, etc.) may require a more complete context tree. The subjects that become contexts are those that cover areas of information which will always, or nearly always, be used to define the value of the root, goal-parameters. The infectious blood disease domain of MYCIN, for example, is described by the following contexts:

PATIENT
CULTURE
ORGANISM

Each of these contexts has multiple parameters leading to the goal-parameter(s) values. The management domain might include contexts such as:



or:



The selection of contexts is strictly the systems designer's choice. The system is best kept as uncomplicated as the subject will permit. Rules and parameters will be grouped by the RULEGROUP and PROPGROUP properties, respectively, and selected by the rule interpreter/monitor for use in the program. The designer's responsibility is to design a logical set of backward-chaining rules, to define all rules and parameters, and to structure the context tree.

B. ARTIFICIAL INTELLIGENCE MOTIVATIONS

The following example and the motivation descriptions for AI work are offered to stimulate further developments in the field.

Artificial Intelligence programs may be described as containing powerful search algorithms (control strategies) operating upon data bases in order to efficiently produce relevant logical inferences. This description mentions algorithms, necessary to all computer work, and the use of data but the energy required to accurately represent data in a "real world" manner must also be emphasized [Sacerdoti, 1979]. Thus the two key issues in AI work are control and data representation.

Representation and control become important when going beyond toy systems and proceeding into more useful programs. A chess game is used as an example. The game can be described as the use of a database consisting of a number of legal moves. This description may describe the game but

does not tell one how to build a better chess program or how to play better chess.

Therefore, two major motivations are established for Artificial Intelligence work. First, analogous to building a better chess program, there is the engineering motivation to make computers more useful by making them "smarter" -- endowing them with common sense, giving them access to a full sensory world, and enabling them to understand humans on the human's terms. Secondly, analogous to playing better chess, there is the scientific motivation to study the nature of knowledge and the mechanisms of intelligent behavior independent of their embodiment in man or machine. As with any computing device or concept, Artificial Intelligence work is designed to extend man's capabilities permitting him to perform more and, hopefully, better accomplishments.

C. STUDY AREAS

Of the time expended to complete this thesis, the major portion (approximately three and one half months) was occupied by learning the EMYCIN system and the supportive concepts and programs. This included a study of predicate calculus as it applies to Artificial Intelligence, the study of production rules, the use of backward-chaining in a goal-seeking system, an overview of the LISP programming language and its functions, and the use of the ARPANET system which permitted the transfer of the EMYCIN file from Stanford University to the University of Southern California. The transfer of

the file, and the resulting, successful portability was facilitated by both schools using DEC-1090 computers with the TOPS-20 operating system. Additionally, familiarity with user features of the TENEX operating system, the HERMES automatic message handling system, the Experimental Editor (XED), and debugging procedures used with the INTERLISP programming language were also required to complete this thesis. Continued work in AI, in particular the DECAIDS program using the ARPANET, will require a review of the areas of study listed.

In order to successfully complete an AI project, such as DECAIDS, additional time must be scheduled for deriving the data to be used in a knowledge base. Coincident to the previous list of study areas was the effort required to collect the required expert knowledge. The modeling of a management system required approximately one and one half month's time. The management modeling process continued into what was informally referred to as the use of the EMYCIN "production rule language."

The lack of user oriented documentation concerning the use of the EMYCIN production rule language caused some delays in completing the knowledge base and producing a running program. While an "EMYCIN.DOC" documentation file has been prepared by the Stanford AI group, it has been written more as a set of very complete notes for the Stanford group rather than as a set of user oriented instructions. Problems in getting the DECAIDS program on-line were resolved by

ARPANET messages, telephone calls, and personal visits to the Stanford group and by reference to the INTERLISP manual. This interaction between NPS and Stanford did assist the efforts of the AI people at Stanford in extending the user information contained in the documentation for EMYCIN.

Acknowledgement for a vast amount of information and selfless assistance in sharing her knowledge of production rules and their application to AI is given to Miss Carlisle Scott of the Stanford University Computer Science Department. Recognition is also given to Dr. E. Fiegenbaum (Chairman of the Computer Science Department at Stanford University) for his assistance, and to Dr. Nils J. Nilsson (Stanford Research International, Menlo Park, California) for permission to use material from his forthcoming book, Artificial Intelligence -- A Framework.

APPENDIX A

DECAIDS PRODUCTION RULES

```
RULE001
-------
[This rule is definitional, applies to ORGANIZATION, and is tried
     when information is received about THE DECISION]

  If:  An attempt has been made to deduce THE DECISION
  Then:  Display THE DECISION


  PREMISE:  ($AND (ONCEKNOWN CNTXT DECAIDS T))
  ACTION:  (PRINTCONCLUSIONS CNTXT DECAIDS T)

[ORGRULES/antecedent]


RULE002
-------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE DECISION]

  If:  1) THE TYPE OF SYSTEM RECOMMENDED TO BE USED is known,
       2) THETYPE OF OUTPUT DEVICE TO BE USED is known,
       3) THE RECOMMENDED INSTALLATION TO BE USED is known, and
       4) THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED is
          known
  Then:  It is definite (1.0) that the following is one of THE
           DECISION: USE THE FOLLOWING TYPE SYSTEM <typsys> THE
           OUTPUT SHOULD BE DISPLAYED ON <output> THE MANNER OF
           INSTALLATION SHOULD BE <install> AND THE RECOMMENDATION
           FOR TRAINING AND ASSISTANCE IS <trg> .


  PREMISE:  ($AND (KNOWN CNTXT TYPSYS)
                  (KNOWN CNTXT OUTPUT)
                  (KNOWN CNTXT INSTALL)
                  (KNOWN CNTXT TRG))
  ACTION:  (CONCLUDETEXT CNTXT DECAIDS (TEXT NIL
                                   'USE THE FOLLOWING TYPE SYSTEM'
                                           (VAL1 CNTXT TYPSYS)

                               'THE OUTPUT SHOULD BE DISPLAYED ON'
                                           (VAL1 CNTXT OUTPUT)

                            'THE MANNER OF INSTALLATION SHOULD BE'
                                           (VAL1 CNTXT INSTALL)

          'AND THE RECOMMENDATION FOR TRAINING AND ASSISTANCE IS'
                                           (VAL1 CNTXT TRG)
                                           '.')
                        TALLY 1000)
```

```
RULE003
--------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
     OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
     USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  THE ORGANIZATION'S STRUCTURE is centralized
  Then:  1) There is strongly suggestive evidence (.8) that THE TYPE
            OF SYSTEM RECOMMENDED TO BE USED is real-time,
         2) There is strongly suggestive evidence (.8) that THETYPE
            OF OUTPUT DEVICE TO BE USED is individual-terminals,
         3) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED INSTALLATION TO BE USED is pyramidal, and
         4) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED is do-
            not-hire-specialists


  PREMISE:  ($AND (SAME CNTXT INFORMAL CENTRALIZED))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS REAL-TIME TALLY 800)
                   (CONCLUDE CNTXT OUTPUT INDIVIDUAL-TERMINALS TALLY
                             800)
                   (CONCLUDE CNTXT INSTALL PYRAMIDAL TALLY 800)
                   (CONCLUDE CNTXT TRG DO-NOT-HIRE-SPECIALISTS TALLY
                             800))

[ORGRULES]


RULE004
--------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
     OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
     USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  THE LEADER'S LEVEL OF TRAINING is skilled
  Then:  1) There is strongly suggestive evidence (.8) that THE TYPE
            OF SYSTEM RECOMMENDED TO BE USED is real-time,
         2) Hire-specialists,
         3) There is strongly suggestive evidence (.8) that THETYPE
            OF OUTPUT DEVICE TO BE USED is individual-terminals,
         4) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED INSTALLATION TO BE USED is pyramidal, and
         5) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED is do-
            not-hire-specialists


  PREMISE:  ($AND (SAME CNTXT LEADER-TRAINING SKILLED))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS REAL-TIME TALLY 800)
                  HIRE-SPECIALISTS
                   (CONCLUDE CNTXT OUTPUT INDIVIDUAL-TERMINALS TALLY
                             800)
                   (CONCLUDE CNTXT INSTALL PYRAMIDAL TALLY 800)
                   (CONCLUDE CNTXT TRG DO-NOT-HIRE-SPECIALISTS TALLY
                             800))

[ORGRULES]
```

```
RULE005
-------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
     OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
     USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  THE LEADER'S LEVEL OF TRAINING is known
  Then:  1) There is suggestive evidence (.5) that THE TYPE OF SYSTEM
            RECOMMENDED TO BE USED is real-time,
         2) There is suggestive evidence (.5) that THETYPE OF OUTPUT
            DEVICE TO BE USED is individual-terminals,
         3) There is suggestive evidence (.5) that THE RECOMMENDED
            INSTALLATION TO BE USED is unavailable, and
         4) There is suggestive evidence (.6) that THE RECOMMENDED
            TRAINING OR ASSISTANCE TO BE ACQUIRED is hire-specialists


  PREMISE:  ($AND (KNOWN CNTXT LEADER-TRAINING UNSKILLED))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS REAL-TIME TALLY 500)
                   (CONCLUDE CNTXT OUTPUT INDIVIDUAL-TERMINALS TALLY
                             500)
                   (CONCLUDE CNTXT INSTALL UNAVAILABLE TALLY 500)
                   (CONCLUDE CNTXT TRG HIRE-SPECIALISTS TALLY 600))

[ORGRULES]


RULE006
-------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
     OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
     USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  THE STAFF'S LEVEL OF TECHNICAL TRAINING is known
  Then:  1) There is suggestive evidence (.5) that THE TYPE OF SYSTEM
            RECOMMENDED TO BE USED is real-time,
         2) There is suggestive evidence (.5) that THETYPE OF OUTPUT
            DEVICE TO BE USED is individual-terminals,
         3) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED INSTALLATION TO BE USED is pyramidal, and
         4) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED is do-
            not-hire-specialists


  PREMISE:  ($AND (KNOWN CNTXT STFFTRG SKILLED))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS REAL-TIME TALLY 500)
                   (CONCLUDE CNTXT OUTPUT INDIVIDUAL-TERMINALS TALLY
                             500)
                   (CONCLUDE CNTXT INSTALL PYRAMIDAL TALLY 800)
                   (CONCLUDE CNTXT TRG DO-NOT-HIRE-SPECIALISTS TALLY
                             800))

[ORGRULES]
```

```
RULE007
-------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
     OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
     USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  THE STAFF'S LEVEL OF TECHNICAL TRAINING is unskilled
  Then:  1) There is strongly suggestive evidence (.8) that THE TYPE
            OF SYSTEM RECOMMENDED TO BE USED is real-time,
         2) There is strongly suggestive evidence (.8) that THETYPE
            OF OUTPUT DEVICE TO BE USED is individual-terminals,
         3) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED INSTALLATION TO BE USED is divisional, and
         4) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED is hire-
            specialists


  PREMISE:  ($AND (SAME CNTXT STFFTRG UNSKILLED))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS REAL-TIME TALLY 800)
                   (CONCLUDE CNTXT OUTPUT INDIVIDUAL-TERMINALS TALLY
                             800)
                   (CONCLUDE CNTXT INSTALL DIVISIONAL TALLY 800)
                   (CONCLUDE CNTXT TRG HIRE-SPECIALISTS TALLY 800))

[ORGRULES]


RULE008
-------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
     OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
     USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  THE LEVEL OF THE TASK'S STRESS is high
  Then:  1) There is strongly suggestive evidence (.8) that THE TYPE
            OF SYSTEM RECOMMENDED TO BE USED is real-time,
         2) There is strongly suggestive evidence (.8) that THETYPE
            OF OUTPUT DEVICE TO BE USED is individual-terminals,
         3) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED INSTALLATION TO BE USED is pyramidal, and
         4) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED is do-
            not-hire-specialists


  PREMISE:  ($AND (SAME CNTXT STRESS HIGH))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS REAL-TIME TALLY 800)
                   (CONCLUDE CNTXT OUTPUT INDIVIDUAL-TERMINALS TALLY
                             800)
                   (CONCLUDE CNTXT INSTALL PYRAMIDAL TALLY 800)
                   (CONCLUDE CNTXT TRG DO-NOT-HIRE-SPECIALISTS TALLY
                             800))

[ORGRULES]
```

```
RULE009
--------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
     OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
     USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  THE LEVEL OF THE TASK'S STRESS is low
  Then:  1) There is suggestive evidence (.5) that THE TYPE OF SYSTEM
            RECOMMENDED TO BE USED is real-time,
         2) There is suggestive evidence (.5) that THETYPE OF OUTPUT
            DEVICE TO BE USED is individual-terminals,
         3) There is suggestive evidence (.5) that THE RECOMMENDED
            INSTALLATION TO BE USED is divisional, and
         4) There is suggestive evidence (.5) that THE RECOMMENDED
            TRAINING OR ASSISTANCE TO BE ACQUIRED is do-not-hire-
            specialists


  PREMISE:  ($AND (SAME CNTXT STRESS LOW))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS REAL-TIME TALLY 500)
                   (CONCLUDE CNTXT OUTPUT INDIVIDUAL-TERMINALS TALLY
                             500)
                   (CONCLUDE CNTXT INSTALL DIVISIONAL TALLY 500)
                   (CONCLUDE CNTXT TRG DO-NOT-HIRE-SPECIALISTS TALLY
                             500))

[ORGRULES]


RULE013
--------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
     OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
     USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  THE TASK DEFINITION is clearly-defined
  Then:  1) There is strongly suggestive evidence (.8) that THE TYPE
            OF SYSTEM RECOMMENDED TO BE USED is real-time,
         2) There is strongly suggestive evidence (.8) that THETYPE
            OF OUTPUT DEVICE TO BE USED is individual-terminals,
         3) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED INSTALLATION TO BE USED is pyramidal, and
         4) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED is do-
            not-hire-specialists


  PREMISE:  ($AND (SAME CNTXT PROBDEF CLEARLY-DEFINED))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS REAL-TIME TALLY 800)
                   (CONCLUDE CNTXT OUTPUT INDIVIDUAL-TERMINALS TALLY
                             800)
                   (CONCLUDE CNTXT INSTALL PYRAMIDAL TALLY 800)
                   (CONCLUDE CNTXT TRG DO-NOT-HIRE-SPECIALISTS TALLY
                             800))

[ORGRULES]
```

```
RULE014
-------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
     OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
     USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  THE TASK DEFINITION is ambiguous
  Then:  1) There is suggestive evidence (.5) that THE TYPE OF SYSTEM
            RECOMMENDED TO BE USED is real-time,
         2) There is suggestive evidence (.5) that THETYPE OF OUTPUT
            DEVICE TO BE USED is individual-terminals,
         3) There is suggestive evidence (.5) that THE RECOMMENDED
            INSTALLATION TO BE USED is divisional, and
         4) There is suggestive evidence (.5) that THE RECOMMENDED
            TRAINING OR ASSISTANCE TO BE ACQUIRED is do-not-hire-
            specialists


  PREMISE:  ($AND (SAME CNTXT PRODDEF AMBIGUOUS))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS REAL-TIME TALLY 500)
                   (CONCLUDE CNTXT OUTPUT INDIVIDUAL-TERMINALS TALLY
                             500)
                   (CONCLUDE CNTXT INSTALL DIVISIONAL TALLY 500)
                   (CONCLUDE CNTXT TRG DO-NOT-HIRE-SPECIALISTS TALLY
                             500))

[ORGRULES]


RULE015
-------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
     OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
     USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  THE ORGANIZATION'S STRUCTURE is centralized
  Then:  1) There is strongly suggestive evidence (.8) that THE TYPE
            OF SYSTEM RECOMMENDED TO BE USED is real-time,
         2) There is strongly suggestive evidence (.8) that THETYPE
            OF OUTPUT DEVICE TO BE USED is individual-terminals,
         3) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED INSTALLATION TO BE USED is pyramidal, and
         4) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED is do-
            not-hire-specialists


  PREMISE:  ($AND (SAME CNTXT INFORMAL CENTRALIZED))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS REAL-TIME TALLY 800)
                   (CONCLUDE CNTXT OUTPUT INDIVIDUAL-TERMINALS TALLY
                             800)
                   (CONCLUDE CNTXT INSTALL PYRAMIDAL TALLY 800)
                   (CONCLUDE CNTXT TRG DO-NOT-HIRE-SPECIALISTS TALLY
                             800))

[ORGRULES]
```

```
RULE018
-------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
     OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
     USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  THE ORGANIZATION'S STRUCTURE is consultative


  Then:  1) There is strongly suggestive evidence (.8) that THE TYPE
            OF SYSTEM RECOMMENDED TO BE USED is real-time,
         2) There is strongly suggestive evidence (.8) that THETYPE
            OF OUTPUT DEVICE TO BE USED is individual-terminals,
         3) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED INSTALLATION TO BE USED is divisional, and
         4) There is suggestive evidence (.5) that THE RECOMMENDED
            TRAINING OR ASSISTANCE TO BE ACQUIRED is do-not-hire-
            specialists


  PREMISE:  ($AND (SAME CNTXT INFORMAL CONSULTATIVE))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS REAL-TIME TALLY 800)
                   (CONCLUDE CNTXT OUTPUT INDIVIDUAL-TERMINALS TALLY
                             800)
                   (CONCLUDE CNTXT INSTALL DIVISIONAL TALLY 800)
                   (CONCLUDE CNTXT TRG DO-NOT-HIRE-SPECIALISTS TALLY
                             500))

[ORGRULES]


RULE019
-------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
     OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
     USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  THE ORGANIZATION'S STRUCTURE is partially-delegated
  Then:  1) There is strongly suggestive evidence (.8) that THE TYPE
            OF SYSTEM RECOMMENDED TO BE USED is real-time,
         2) There is strongly suggestive evidence (.8) that THETYPE
            OF OUTPUT DEVICE TO BE USED is individual-terminals,
         3) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED INSTALLATION TO BE USED is divisional, and
         4) There is suggestive evidence (.5) that THE RECOMMENDED
            TRAINING OR ASSISTANCE TO BE ACQUIRED is hire-specialists


  PREMISE:  ($AND (SAME CNTXT INFORMAL PARTIALLY-DELEGATED))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS REAL-TIME TALLY 800)
                   (CONCLUDE CNTXT OUTPUT INDIVIDUAL-TERMINALS TALLY
                             800)
                   (CONCLUDE CNTXT INSTALL DIVISIONAL TALLY 800)
                   (CONCLUDE CNTXT TRG HIRE-SPECIALISTS TALLY 500))

[ORGRULES]
```

```
RULE016
-------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
     OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
     USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  THE ORGANIZATION'S STRUCTURE is line
  Then:  1) There is suggestive evidence (.5) that THE TYPE OF SYSTEM
            RECOMMENDED TO BE USED is real-time,
         2) There is suggestive evidence (.5) that THETYPE OF OUTPUT
            DEVICE TO BE USED is individual-terminals,
         3) There is suggestive evidence (.5) that THE RECOMMENDED
            INSTALLATION TO BE USED is pyramidal, and
         4) There is suggestive evidence (.5) that THE RECOMMENDED
            TRAINING OR ASSISTANCE TO BE ACQUIRED is hire-specialists


  PREMISE:  ($AND (SAME CNTXT FORMAL LINE))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS REAL-TIME TALLY 500)
                   (CONCLUDE CNTXT OUTPUT INDIVIDUAL-TERMINALS TALLY
                             500)
                   (CONCLUDE CNTXT INSTALL PYRAMIDAL TALLY 500)
                   (CONCLUDE CNTXT TRG HIRE-SPECIALISTS TALLY 500))

[ORGRULES]


RULE017
-------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
     OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
     USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  THE ORGANIZATION'S STRUCTURE is THE FORMAL COMPOSITION OF THE
          ORGANIZATION'S STRUCTURE
  Then:  1) There is strongly suggestive evidence (.8) that THE TYPE
            OF SYSTEM RECOMMENDED TO BE USED is real-time,
         2) There is strongly suggestive evidence (.8) that THETYPE
            OF OUTPUT DEVICE TO BE USED is individual-terminals,
         3) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED INSTALLATION TO BE USED is pyramidal, and
         4) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED is do-
            not-hire-specialists


  PREMISE:  ($AND (SAME CNTXT FORMAL STAFF))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS REAL-TIME TALLY 800)
                   (CONCLUDE CNTXT OUTPUT INDIVIDUAL-TERMINALS TALLY
                             800)
                   (CONCLUDE CNTXT INSTALL PYRAMIDAL TALLY 800)
                   (CONCLUDE CNTXT TRG DO-NOT-HIRE-SPECIALISTS TALLY
                             800))

[ORGRULES]
```

```
RULE022
-------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
     OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
     USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  THE ORGANIZATION'S STRUCTURE is decentralized
  Then:  1) There is strongly suggestive evidence (.8) that THE TYPE
            OF SYSTEM RECOMMENDED TO BE USED is real-time,
         2) There is strongly suggestive evidence (.8) that THETYPE
            OF OUTPUT DEVICE TO BE USED is individual-terminals,
         3) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED INSTALLATION TO BE USED is divisional, and
         4) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED is hire-
            specialists


  PREMISE:  ($AND (SAME CNTXT INFORMAL DECENTRALIZED))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS REAL-TIME TALLY 800)
                   (CONCLUDE CNTXT OUTPUT INDIVIDUAL-TERMINALS TALLY
                             800)
                   (CONCLUDE CNTXT INSTALL DIVISIONAL TALLY 800)
                   (CONCLUDE CNTXT TRG HIRE-SPECIALISTS TALLY 800))

[ORGRULES]


RULE023
-------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
     OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
     USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  THE LEADER'S STYLE OF OPERATION is task-oriented
  Then:  1) There is strongly suggestive evidence (.8) that THE TYPE
            OF SYSTEM RECOMMENDED TO BE USED is real-time,
         2) There is strongly suggestive evidence (.8) that THETYPE
            OF OUTPUT DEVICE TO BE USED is individual-terminals,
         3) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED INSTALLATION TO BE USED is pyramidal, and
         4) There is suggestive evidence (.5) that THE RECOMMENDED
            TRAINING OR ASSISTANCE TO BE ACQUIRED is unavailable


  PREMISE:  ($AND (SAME CNTXT STYLE TASK-ORIENTED))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS REAL-TIME TALLY 800)
                   (CONCLUDE CNTXT OUTPUT INDIVIDUAL-TERMINALS TALLY
                             800)
                   (CONCLUDE CNTXT INSTALL PYRAMIDAL TALLY 800)
                   (CONCLUDE CNTXT TRG UNAVAILABLE TALLY 500))

[ORGRULES]
```

```
RULE024
--------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
     OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
     USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  THE INDIVIDUAL'S TECHNICAL TRAINING IN DECISION ANALYSIS is
         high
  Then:  1) There is strongly suggestive evidence (.8) that THE TYPE
            OF SYSTEM RECOMMENDED TO BE USED is real-time,
         2) There is strongly suggestive evidence (.8) that THETYPE
            OF OUTPUT DEVICE TO BE USED is individual-terminals,
         3) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED INSTALLATION TO BE USED is pyramidal, and
         4) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED is
            train-existing-staff


  PREMISE:  ($AND (SAME CNTXT TRAINING HIGH))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS REAL-TIME TALLY 800)
                   (CONCLUDE CNTXT OUTPUT INDIVIDUAL-TERMINALS TALLY
                             800)
                   (CONCLUDE CNTXT INSTALL PYRAMIDAL TALLY 800)
                   (CONCLUDE CNTXT TRG TRAIN-EXISTING-STAFF TALLY 800))

[ORGRULES]


RULE025
--------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
     OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
     USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  THE INDIVIDUAL'S TECHNICAL TRAINING IN DECISION ANALYSIS is
         low
  Then:  1) There is strongly suggestive evidence (.8) that THE TYPE
            OF SYSTEM RECOMMENDED TO BE USED is non-real-time,
         2) There is strongly suggestive evidence (.8) that THETYPE
            OF OUTPUT DEVICE TO BE USED is large-screen-displays,
         3) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED INSTALLATION TO BE USED is divisional, and
         4) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED is hire-
            specialists


  PREMISE:  ($AND (SAME CNTXT TRAINING LOW))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS NON-REAL-TIME TALLY 800)
                   (CONCLUDE CNTXT OUTPUT LARGE-SCREEN-DISPLAYS TALLY
                             800)
                   (CONCLUDE CNTXT INSTALL DIVISIONAL TALLY 800)
                   (CONCLUDE CNTXT TRG HIRE-SPECIALISTS TALLY 800))

[ORGRULES]
```

```
RULE026
-------
[This rule applies to ORGANIZATION, and is tried in order to find out
    about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
    OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
    USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  THE TECHNOLOGICAL METHODS AVAILABLE is analytical-aids


  Then:  1) There is strongly suggestive evidence (.8) that THE TYPE
            OF SYSTEM RECOMMENDED TO BE USED is real-time,
         2) There is strongly suggestive evidence (.8) that THETYPE
            OF OUTPUT DEVICE TO BE USED is individual-terminals,
         3) There is suggestive evidence (.5) that THE RECOMMENDED
            INSTALLATION TO BE USED is pyramidal, and
         4) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED is
            unavailable

  PREMISE:  ($AND (SAME CNTXT METHODS ANALYTICAL-AIDS))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS REAL-TIME TALLY 800)
                   (CONCLUDE CNTXT OUTPUT INDIVIDUAL-TERMINALS TALLY
                             800)
                   (CONCLUDE CNTXT INSTALL PYRAMIDAL TALLY 500)
                   (CONCLUDE CNTXT TRG UNAVAILABLE TALLY 800))

[ORGRULES]

RULE027
-------
[This rule applies to ORGANIZATION, and is tried in order to find out
    about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
    OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
    USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  THE TECHNOLOGICAL METHODS AVAILABLE is inventory-aids
  Then:  1) There is suggestive evidence (.5) that THE TYPE OF SYSTEM
            RECOMMENDED TO BE USED is real-time,
         2) There is suggestive evidence (.5) that THETYPE OF OUTPUT
            DEVICE TO BE USED is individual-terminals,
         3) There is suggestive evidence (.5) that THE RECOMMENDED
            INSTALLATION TO BE USED is pyramidal, and
         4) There is suggestive evidence (.5) that THE RECOMMENDED
            TRAINING OR ASSISTANCE TO BE ACQUIRED is unavailable

  PREMISE:  ($AND (SAME CNTXT METHODS INVENTORY-AIDS))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS REAL-TIME TALLY 500)
                   (CONCLUDE CNTXT OUTPUT INDIVIDUAL-TERMINALS TALLY
                             500)
                   (CONCLUDE CNTXT INSTALL PYRAMIDAL TALLY 500)
                   (CONCLUDE CNTXT TRG UNAVAILABLE TALLY 500))

[ORGRULES]
```

```
RULE028
-------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
     OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
     USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  The technological knowledge-level is skilled
  Then:  1) There is suggestive evidence (.5) that THE TYPE OF SYSTEM
            RECOMMENDED TO BE USED is unavailable,
         2) There is suggestive evidence (.5) that THETYPE OF OUTPUT
            DEVICE TO BE USED is unavailable,
         3) There is suggestive evidence (.5) that THE RECOMMENDED
            INSTALLATION TO BE USED is unavailable, and
         4) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED is do-
            not-hire-specialists


  PREMISE:  ($AND (SAME CNTXT KNOWLEDGE-LEVEL SKILLED))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS UNAVAILABLE TALLY 500)
                   (CONCLUDE CNTXT OUTPUT UNAVAILABLE TALLY 500)
                   (CONCLUDE CNTXT INSTALL UNAVAILABLE TALLY 500)
                   (CONCLUDE CNTXT TRG DO-NOT-HIRE-SPECIALISTS TALLY
                             800))

[ORGRULES]


RULE029
-------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
     OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
     USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  The technological knowledge-level is unskilled
  Then:  1) There is strongly suggestive evidence (.8) that THE TYPE
            OF SYSTEM RECOMMENDED TO BE USED is unavailable,
         2) There is strongly suggestive evidence (.8) that THETYPE
            OF OUTPUT DEVICE TO BE USED is unavailable,
         3) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED INSTALLATION TO BE USED is unavailable, and
         4) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED is hire-
            specialists


  PREMISE:  ($AND (SAME CNTXT KNOWLEDGE-LEVEL UNSKILLED))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS UNAVAILABLE TALLY 800)
                   (CONCLUDE CNTXT OUTPUT UNAVAILABLE TALLY 800)
                   (CONCLUDE CNTXT INSTALL UNAVAILABLE TALLY 800)
                   (CONCLUDE CNTXT TRG HIRE-SPECIALISTS TALLY 800))

[ORGRULES]
```

```
RULE030
-------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
     OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
     USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  THE ORGANIZATION'S STRUCTURE is line
  Then:  1) There is strongly suggestive evidence (.8) that THE TYPE
            OF SYSTEM RECOMMENDED TO BE USED is real-time,
         2) There is strongly suggestive evidence (.8) that THETYPE
            OF OUTPUT DEVICE TO BE USED is unavailable,
         3) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED INSTALLATION TO BE USED is pyramidal, and
         4) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED is
            unavailable


  PREMISE:  ($AND (SAME CNTXT FORMAL LINE))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS REAL-TIME TALLY 800)
                   (CONCLUDE CNTXT OUTPUT UNAVAILABLE TALLY 800)
                   (CONCLUDE CNTXT INSTALL PYRAMIDAL TALLY 800)
                   (CONCLUDE CNTXT TRG UNAVAILABLE TALLY 800))

[ORGRULES]


RULE031
-------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
     OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
     USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  THE ORGANIZATION'S STRUCTURE is functional
  Then:  1) There is strongly suggestive evidence (.8) that THE TYPE
            OF SYSTEM RECOMMENDED TO BE USED is non-real-time,
         2) There is strongly suggestive evidence (.8) that THETYPE
            OF OUTPUT DEVICE TO BE USED is unavailable,
         3) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED INSTALLATION TO BE USED is divisional, and
         4) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED is
            unavailable


  PREMISE:  ($AND (SAME CNTXT FORMAL FUNCTIONAL))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS NON-REAL-TIME TALLY 800)
                   (CONCLUDE CNTXT OUTPUT UNAVAILABLE TALLY 800)
                   (CONCLUDE CNTXT INSTALL DIVISIONAL TALLY 800)
                   (CONCLUDE CNTXT TRG UNAVAILABLE TALLY 800))

[ORGRULES]
```

```
RULE032
-------
[This rule applies to ORGANIZATION, and is tried in order to find out
    about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
    OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
    USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  THE ORGANIZATION'S STRUCTURE is matrix
  Then:  1) There is strongly suggestive evidence (.8) that THE TYPE
            OF SYSTEM RECOMMENDED TO BE USED is non-real-time,
         2) There is strongly suggestive evidence (.8) that THETYPE
            OF OUTPUT DEVICE TO BE USED is unavailable,
         3) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED INSTALLATION TO BE USED is divisional, and
         4) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED is
            unavailable


  PREMISE:  ($AND (SAME CNTXT FORMAL MATRIX))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS NON-REAL-TIME TALLY 800)
                   (CONCLUDE CNTXT OUTPUT UNAVAILABLE TALLY 800)
                   (CONCLUDE CNTXT INSTALL DIVISIONAL TALLY 800)
                   (CONCLUDE CNTXT TRG UNAVAILABLE TALLY 800))

[ORGRULES]


RULE033
-------
[This rule applies to ORGANIZATION, and is tried in order to find out
    about the recommended informal structure]

  If:  THE ORGANIZATION'S STRUCTURE is notavailable
  Then:  There is strongly suggestive evidence (.8) that the
            recommended informal structure is centralized


  PREMISE:  ($AND (SAME CNTXT INFORMAL NOTAVAILABLE))
  ACTION:  (CONCLUDE CNTXT UNKSTRUC CENTRALIZED TALLY 800)

[ORGRULES]


RULE034
-------
[This rule applies to ORGANIZATION, and is tried in order to find out
    about the recommended informal structure]

  If:  The recommended informal structure is known
  Then:  It is definite (1.0) that the following is the recommended
            informal structure: THE
            RECOMMENDED INFORMAL STRUCTURE TO USE IS <unkstruc>


  PREMISE:  ($AND (KNOWN CNTXT UNKSTRUC))
  ACTION:  (CONCLUDETEXT CNTXT UNKSTRUC (TEXT NIL
                    'THE
RECOMMENDED INFORMAL STRUCTURE TO USE IS'
                                             (VAL1 CNTXT UNKSTRUC))
                          TALLY 1000)
```

```
RULE035
-------
[This rule is definitional, applies to ORGANIZATION, and is tried
     when information is received about the recommended informal
     structure]

  If:  An attempt has been made to deduce the recommended informal
           structure
  Then:  Display the recommended informal structure


  PREMISE:  ($AND (ONCEKNOWN CNTXT UNKSTRUC t))
  ACTION:  (PRINTCONCLUSIONS CNTXT UNKSTRUC)

[ORGRULES/antecedent]


RULE036
-------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE TYPE OF SYSTEM RECOMMENDED TO BE USED, THETYPE OF
     OUTPUT DEVICE TO BE USED, THE RECOMMENDED INSTALLATION TO BE
     USED or THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED]

  If:  THE IMPLEMENTATION/CONSTRUCTION STATUS OF THE SYSTEM
  Then:  1) There is strongly suggestive evidence (.8) that THE TYPE
            OF SYSTEM RECOMMENDED TO BE USED is real-time,
         2) There is strongly suggestive evidence (.8) that THETYPE
            OF OUTPUT DEVICE TO BE USED is individual-terminals,
         3) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED INSTALLATION TO BE USED is pyramidal, and
         4) There is strongly suggestive evidence (.8) that THE
            RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED is
            train-existing-staff


  PREMISE:  ($AND (SAME CNTXT SYSSTAT YES))
  ACTION:  (DO-ALL (CONCLUDE CNTXT TYPSYS REAL-TIME TALLY 800)
                   (CONCLUDE CNTXT OUTPUT INDIVIDUAL-TERMINALS TALLY
                             800)
                   (CONCLUDE CNTXT INSTALL PYRAMIDAL TALLY 800)
                   (CONCLUDE CNTXT TRG train-existing-staff TALLY 800))

[ORGRULES]


RULE039
-------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE RECOMMENDED FORMAL STRUCTURE]

  If:  THE ORGANIZATION'S STRUCTURE is notavailable
  Then:  There is strongly suggestive evidence (.8) that THE
            RECOMMENDED FORMAL STRUCTURE is line


  PREMISE:  ($AND (SAME CNTXT FORMAL NOTAVAILABLE))
  ACTION:  (CONCLUDE CNTXT FORSTRUC LINE TALLY 800)
```

```
[ORGRULES]


RULE040
-------
[This rule applies to ORGANIZATION, and is tried in order to find out
     about THE RECOMMENDED FORMAL STRUCTURE]

  If:  THE RECOMMENDED FORMAL STRUCTURE is known
  Then:  It is definite (1.0) that the following is THE RECOMMENDED
            FORMAL STRUCTURE: THE RECOMMENDED FORMAL STRUCTURE TO USE
            IS <forstruc>


  PREMISE:  ($AND (KNOWN CNTXT FORSTRUC))
  ACTION:  (CONCLUDETEXT CNTXT FORSTRUC (TEXT NIL
                        'THE RECOMMENDED FORMAL STRUCTURE TO USE IS'
                                              (VAL1 CNTXT FORSTRUC))
                          TALLY 1000)

[ORGRULES]


RULE041
-------
[This rule is definitional, applies to ORGANIZATION, and is tried
     when information is received about THE RECOMMENDED FORMAL
     STRUCTURE]

  If:  An attempt has been made to deduce THE RECOMMENDED FORMAL
          STRUCTURE
  Then:  Display THE RECOMMENDED FORMAL STRUCTURE


  PREMISE:  ($AND (ONCEKNOWN CNTXT FORSTRUC t))
  ACTION:  (PRINTCONCLUSIONS CNTXT FORSTRUC)

[ORGRULES/antecedent]

NIL
-
```

APPENDIX B

DECAIDS PARAMETER LISTING

```
    DECAIDS  ORG                METHODS  ORG                STRESS  ORG
    ENVIRONMENT  ORG            ORGANIZATION  VAL           STYLE  ORG
    FORMAL  ORG                 ORGRULES  ALLNAMES          SYSSTAT  ORG
    FORSTRUC  ORG               OUTPUT  ORG                 TRAINING  ORG
    INFORMAL  ORG               PROBDEF  ORG                TRG  ORG
    INSTALL  ORG                PROBTYPE  ORG               TYPSYS  ORG
    KNOWLEDGE-LEVEL  ORG        STAFF  ORG                  UNKSTRUC  ORG
    LEADER-TRAINING  ORG        STFFTRG  ORG
                                   PROP-ORG
                                   --------

DECAIDS
-------
  ANTECEDENT-IN:  (RULE001)
  UPDATED-BY-THE-WAY:  (RULE001)
  UPDATED-BY:  (RULE002)
  TRANS:  (THE DECISION)
  LEGALVALS:  TEXT
  MULTIVALUED:  T

ENVIRONMENT
-----------
  TRANS:  (*'s ENVIRONMENT)
  PROMPT:  (WILL YOU COMMENT ON THE LEADER-TRAINING OR THE
                STAFF-TRAINING OF THE ENVIRONMENT OF THE ORGANIZATION?)
  EXPECT:  (LEADER-TRAINING STAFF-TRAINING)
  LABDATA:  T

FORMAL
------
  USED-BY:  (Rules 39 32 31 30 17 16)
  TRANS:  (THE ORGANIZATION'S STRUCTURE)
  PROMPT:  (THE FORMAL STRUCTURE OF THE ORGANIZATION CAN BE DEFINED AS
               EITHER LINE, STAFF, MATRIX, FUNCTIONAL, OR
               NOTAVAILABLE. IF FURTHER EXPLANATION OF THESE TERMS IS
               NEEDED, TYPE A QUESTION MARK. WHAT IS THE
               ORGANIZATION'S FORMAL STRUCTURE?)
  EXPECT:  (LINE STAFF FUNCTIONAL MATRIX NOTAVAILABLE)
  LABDATA:  T
  REPROMPT:  (LINE - EMPHASIZES DIRECT CHAINS OF AUTHORITY AND UNITY OF
                  COMMAND STAFF - INCLUDES AN INFORMATIONAL AND
                  ADVISORY STAFF TO ASSIST AND GUIDE OPERATIONAL
                  PERSONNEL FUNCTIONAL - ARRANGES PERSONNEL BY
                  FUNCTIONAL ACTIVITY SUCH AS LOGISTICS,
                  COMMUNICATIONS, ETC. MATRIX - DRAWS PERSONNEL FROM
                  ACROSS DEPARTMENTAL LINES)

FORSTRUC
--------
  UPDATED-BY:  (Rules 39 SREFMARK 40)
  USED-BY:  (RULE040)
  CONTAINED-IN:  (RULE040)
  ANTECEDENT-IN:  (RULE041)
  UPDATED-BY-THE-WAY:  (RULE041)
  TRANS:  (THE RECOMMENDED FORMAL STRUCTURE)
  EXPECT:  (LINE STAFF FUNCTIONAL MATRIX)
```

```
INFORMAL
--------
  USED-BY:  (Rules 33 22 21 19 18 15 3)
  TRANS:  (THE ORGANIZATION'S STRUCTURE)
  PROMPT:  (THE INFORMAL STRUCTURE OF THE ORGANIZATION REFERS TO THE
                MANNER IN WHICH COMMUNICATION IS ACCOMPLISHED. IS THE
                INFORMAL STRUCTURE BEST DESCRIBED AS CENTRALIZED,
                CONSULTATIVE, TRANSACTIONAL, PARTIALLY-DELEGATED,
                DECENTRALIZED, OR NOTAVAILABLE ? IF FURTHER EXPLANATION
                IS NEEDED, TYPE A QUESTION MARK.)
  EXPECT:  (CENTRALIZED CONSULTATIVE TRANSACTIONAL PARTIALLY-
               DELEGATED DECENTRALIZED NOTAVAILABLE)
  LABDATA:  T
  REPROMPT:  (CENTRALIZED - USES A FOCUSED FLOW OF AUTHORITY TO A
                 SINGLE SOURCE AT THE TOP OF THE HIERARCHY
                 CONSULTATIVE - MAXIMIZES PATTERNS OF CENTRAL CONTROL
                 BUT ENCOURAGES VERTICAL AND UPWARD COMMUNICATION OF
                 ADVICE AND GUIDANCE FROM A PROFESSIONAL STAFF
                 TRANSACTIONAL - STRESSES OPEN COMMUNICATION,
                 DELIBERATION, AND NEGOTIATION, BOTH LATERALLY WITHIN
                 LEVELS AND VERTICALLY AMONG LEVELS.  AUTHORITY MAY
                 STILL REMAIN AT/WITH TOP MANAGEMENT PARTIALLY-
                 DELEGATED - DISTRIBUTES AUTHORITY AMONG PROFESSIONAL
                 STAFF WHILE INCREASING THE NEED FOR CO-ORDINATION OF
                 EFFORT.  THE STAFF MAY POSSESS AUTHORITY TO DEVELOP
                 ACTION ALTERNATIVES BUT TOP MANAGEMENT STILL RETAINS
                 THE RIGHT TO REJECT AND MODIFY DECENTRALIZED -
                 DELEGATES AND DISPERSES FULL DECISION-MAKING POWER
                 TO STAFF AT LOWER LEVELS OF THE HIERARCHY)

INSTALL
-------
  USED-BY:  (RULE002)
  CONTAINED-IN:  (RULE002)
  UPDATED-BY:  (Rules 36 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18
                   17 16 15 14 13 9 8 7 6 5 4 3)
  TRANS:  (THE RECOMMENDED INSTALLATION TO BE USED)
  EXPECT:  (PYRAMIDAL DIVISIONAL UNAVAILABLE)
  REPROMPT:  (A divisional installation places authority in each
                division for independent systems while pyramidal
                installations place authorityat the top of one
                super-system above all divisions.)

KNOWLEDGE-LEVEL
---------------
  USED-BY:  (Rules 29 28)
  TRANS:  (the technological knowledge-level)
  PROMPT:  (IN REGARDS TO THE TECHNOLOGICAL TRAINING REQUIRED TO
               ACCOMPLISH THE TASK, IS THE LEVEL OF TECHNICAL TRAINING
               CONSIDERED TO BE SKILLED, UNSKILLED, OR UNKNOWN?)
  EXPECT:  (SKILLED UNSKILLED)
  LABDATA:  T

LEADER-TRAINING
---------------
  USED-BY:  (Rules 5 4)
  TRANS:  (THE LEADER'S LEVEL OF TRAINING)
  PROMPT:  (Is the task leader's technical training considered to be
               skilled, unskilled, or unknown?)
  EXPECT:  (SKILLED UNSKILLED)
  LABDATA:  T
```

```
METHODS
-------
  USED-BY:  (Rules 27 26)
  TRANS:  (THE TECHNOLOGICAL METHODS AVAILABLE)
  PROMPT:  (ARE THE TECHNOLOGICAL METHODS USED ANALYTICAL-AIDS,
                INVENTORY-AIDS, OR UNKNOWN?)
  EXPECT:  (ANALYTICAL-AIDS INVENTORY-AIDS)
  LABDATA:  T
  REPROMPT:  (Inventory aids refer to administrative uses and
                  analytical aids are those which concern scientific
                  applications.)
                                   ⋮
OUTPUT
------
  USED-BY:  (RULE002)
  CONTAINED-IN:  (RULE002)
  UPDATED-BY:  (Rules 36 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18
                    17 16 15 14 13 9 8 7 6 5 4 3)
  TRANS:  (the type of output device to be used)
  EXPECT:  (INDIVIDUAL-TERMINALS LARGE-SCREEN-DISPLAYS UNAVAILABLE)

PROBDEF
-------
  USED-BY:  (Rules 14 13)
  TRANS:  (THE TASK DEFINITION)
  PROMPT:  (CONCERNING THE PROBLEM FACING THE ORGANIZATION, IS THE
              PROBLEM CLEARLY-DEFINED, AMBIGUOUS, OR UNKNOWN?)
  EXPECT:  (CLEARLY-DEFINED AMBIGUOUS)
  LABDATA:  T

PROBTYPE
--------
  TRANS:  (THE TYPE OF PROBLEM TO BE SOLVED)
  PROMPT:  (THIS PROGRAM IS DESIGNED TO PROVIDE MANAGERS AT ALL LEVELS
                WITH ADVICE CONCERNING THE USE OF THEIR COMPUTER
                RESOURCES. IN ORDER TO PROVIDE THIS INFORMATION, THE
                USER WILL BE ASKED TO FURNISH DATA CONCERNING THE
                ORGANIZATION, ITS LEVEL OF TRAINING, THE
                ORGANIZATION'S LEADER, THE ENVIRONMENT AFFECTING THE
                DECISION, AND THE TASK FACING THE ORGANIZATION. WHAT
                IS THE TYPE OF PROBLEM WHICH THE ORGANIZATION FACES?)
  EXPECT:  ANY
  LABDATA:  T

STAFF
-----
  TRANS:  (THE FORMAL COMPOSITION OF THE ORGANIZATION'S STRUCTURE)
  PROMPT:  (IS THE STAFF'S TECHNICAL TRAINING SKILLED, UNSKILLED, OR
              UNKNOWN?)
  EXPECT:  (SKILLED UNSKILLED)
  LABDATA:  T

STFFTRG
-------
  USED-BY:  (Rules 7 6)
  TRANS:  (THE STAFF'S LEVEL OF TECHNICAL TRAINING)
  PROMPT:  (IS THE ORGANIZATION'S STAFF'S LEVEL OF TECHNICAL TRAINING
              IN THE USE OF COMPUTERIZED TECHNICAL AIDS CONSIDERED
              SKILLED, UNSKILLED, OR UNKNOWN?)
  EXPECT:  (SKILLED UNSKILLED)
  LABDATA:  T
```

```
STRESS
------
  USED-BY:  (Rules 9 8)
  TRANS:  (THE LEVEL OF THE TASK'S STRESS)
  PROMPT:  (IS THE STRESS LEVEL CONSIDERED TO BE HIGH, LOW, OR UNKNOWN?)
  EXPECT:  (HIGH LOW)
  LABDATA:  T

STYLE
-----
  USED-BY:  (Rules 23 20)
  TRANS:  (THE LEADER'S STYLE OF OPERATION)
  PROMPT:  (IS THE TASK LEADER'S STYLE BEST DESCRIBED AS
               RELATION-ORIENTED,TASK-ORIENTED, OR UNKNOWN?)
  EXPECT:  (RELATION-ORIENTED TASK-ORIENTED)
  LABDATA:  T
  REPROMPT:  (relation oriented refers to the leader who gives little
                  direction to his staff, encourages the staff to
                  actively participate in setting decision making
                  parameters, and values the development of personnel
                  responsibility.  Task oriented leaders are defined
                  as those who prefer far more centralization of
                  control and are less concerned with the development
                  of individual responsibility in the decision making
                  process.)

SYSSTAT
-------
  USED-BY:  (RULE036)
  TRANS:  (THE IMPLEMENTATION/CONSTRUCTION STATUS OF THE SYSTEM)
  PROMPT:  (DOES AN OPERATIONAL SYSTEM CURRENTLY EXIST?)
  EXPECT:  (YES NO)
  LABDATA:  T

TRAINING
--------
  USED-BY:  (Rules 25 24)
  TRANS:  (THE INDIVIDUAL'S TECHNICAL TRAINING IN DECISION ANALYSIS)
  PROMPT:  (IS THE TASK LEADER'S LEVEL OF TECHNICAL TRAINING CONSIDERED
               TO BE HIGH, LOW, OR UNKNOWN?)
  EXPECT:  (HIGH LOW)

TRG
---
  USED-BY:  (RULE002)
  CONTAINED-IN:  (RULE002)
  UPDATED-BY:  (Rules 36 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18
                   17 16 15 14 13 9 8 7 6 5 4 3)
  TRANS:  (THE RECOMMENDED TRAINING OR ASSISTANCE TO BE ACQUIRED)
  EXPECT:  (HIRE-SPECIALISTS DO-NOT-HIRE-SPECIALISTS TRAIN-EXISTING-
               STAFF UNAVAILABLE)

TYPSYS
------
  USED-BY:  (RULE002)
  CONTAINED-IN:  (RULE002)
  UPDATED-BY:  (Rules 36 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18
                   17 16 15 14 13 9 8 7 6 5 4 3)
  TRANS:  (THE TYPE OF SYSTEM RECOMMENDED TO BE USED)
  EXPECT:  (REAL-TIME NON-REAL-TIME UNAVAILABLE)
```

```
UNKSTRUC
---------
  UPDATED-BY:  (Rules 33 SREFMARK 34)
  USED-BY:  (RULE034)
  CONTAINED-IN:  (RULE034)
  ANTECEDENT-IN:  (RULE035)
  UPDATED-BY-THE-WAY:  (RULE035)
  TRANS:  (the recommended informal structure)
  EXPECT:  (CENTRALIZED DECENTRALIZED CONSULTATIVE TRANSACTIONAL
                PARTIALLY-DELEGATED)
^LPROP-VAL
^XPROP-VAL
                                        PROP-VAL
                                        --------

ORGANIZATION
------------
  TRANS:  (THE ORGANIZATION)
  MAINPROPS:  (PROBTYPE FORMAL INFORMAL STFFTRG LEADER-TRAINING ROLE
                   PROBDEF KNOWLEDGE-LEVEL STRESS SYSSTAT METHODS)
  PROPTYPE:  PROP-ORG
  TYPE:  ORGANIZATION-
  RULETYPES:  (ORGRULES)
  GOALS:  (DECAIDS UNKSTRUC FORSTRUC)
^LALLNAMES
^XALLNAMES
                                        ALLNAMES
                                        --------

ORGRULES
--------
  CONTEXT:  (ORGANIZATION)
  SVAL:  (ORGANIZATION)
  CTRANS:  ORGANIZATION
NIL
_
```

APPENDIX C

SAMPLE CONSULTATION

```
 BEGIN)
Special options (type ? for help):
** FT 4
Instructions? (Y or N)
** NO

   ---------ORGANIZATION-2---------
   1) THIS PROGRAM IS DESIGNED TO PROVIDE MANAGERS AT ALL LEVELS WITH
      ADVICE CONCERNING THE USE OF THEIR COMPUTER RESOURCES.  IN
      ORDER TO PROVIDE THIS INFORMATION, THE USER WILL BE ASKED TO
      FURNISH DATA CONCERNING: HIS ORGANIZATION, ITS LEVEL OF
      TRAINING, THE ORGANIZATTON'S LEADER, THE ENVIRONMENT AFFECTING
      THE DECISION, AND THE TASK FACING THE ORGANIZATION.  WHAT IS
      THE TYPE OF PROBLEM WHICH THE ORGANIZATION FACES?
   ** THESIS
   2) THE FORMAL STRUCTURE OF THE ORGANIZATION CAN BE DEFINED AS
      EITHER LINE, STAFF, MATRIX, FUNCTIONAL, OR NOTAVAILABLE.  IF
      FURTHER EXPLANATION OF THESE TERMS IS NEEDED, TYPE A QUESTION
      MARK.  WHAT IS THE ORGANIZATION'S FORMAL STRUCTURE?
   ** NOTAVAILABLE
   3) THE INFORMAL STRUCTURE OF THE ORGANIZATION REFERS TO THE MANNER
      IN WHICH COMMUNICATION IS ACCOMPLISHED.  IS THE INFORMAL
      STRUCTURE BEST DESCRIBED AS CENTRALIZED, CONSULTATIVE,
      TRANSACTIONAL, PARTIALLY-DELEGATED, DECENTRALIZED, OR
      NOTAVAILABLE? IF FURTHER EXPLANATION IS NEEDED, TYPE A QUESTION
      MARK.
   ** NOTAVAILABLE
   4) IS THE ORGANIZATTON'S STAFF'S LEVEL OF TECHNICAL TRAINING IN
      THE USE OF COMPUTERIZED TECHNICAL AIDS CONSIDERED SKILLED,
      UNSKILLED, OR UNKNOWN?
   ** SKILLED
   5) Is the task leader's technical training considered to be
      skilled, unskilled, or unknown?
   ** SKILLED
   6) IS THE TASK LEADER'S STYLE BEST DESCRIBED AS RELATION-
      ORIENTED,TASK-ORIENTED, OR UNKNOWN?
   ** RELATION-ORIENTED
   7) CONCERNING THE PROBLEM FACING THE ORGANIZATION, IS THE PROBLEM
      CLEARLY-DEFINED, AMBIGUOUS, OR UNKNOWN?
   ** CLEARLY-DEFINED
   8) IN REGARDS TO THE TECHNOLOGICAL TRAINING REQUIRED TO ACCOMPLISH
      THE TASK, IS THE LEVEL OF TECHNICAL TRAINING CONSIDERED TO BE
      SKILLED, UNSKILLED, OR UNKNOWN?
   ** SKILLED
   9) IS THE STRESS LEVEL CONSIDERED TO BE HIGH, LOW, OR UNKNOWN?
   ** HIGH
   10) DOES AN OPERATIONAL SYSTEM CURRENTLY EXIST?
   ** NO
   11) ARE THE TECHNOLOGICAL METHODS USED ANALYTICAL-AIDS, INVENTORY-
       AIDS, OR UNKNOWN?
   ** ANALYTICAL-AIDS
   --[1] Findout: DECAIDS of ORGANIZATION-2
Trying RULE002/ORGANIZATION-2;
   --[2] Findout: TYPSYS of ORGANIZATION-2
RULE036 failed (in preview) due to clause 1
RULE032 failed (in preview) due to clause 1
RULE031 failed (in preview) due to clause 1
```

```
RULE030 failed {in preview} due to clause 1
RULE029 failed {in preview} due to clause 1
Trying RULE028/ORGANIZATION-2; RULE028 succeeded.
Conclude: TYPSYS of ORGANIZATION-2 is UNAVAILABLE (.5)
Conclude: OUTPUT of ORGANIZATION-2 is UNAVAILABLE (.5)
Conclude: INSTALL of ORGANIZATION-2 is UNAVAILABLE (.5)
Conclude: TRG of ORGANIZATION-2 is DO-NOT-HIRE-SPECIALISTS (.8)
RULE027 failed {in preview} due to clause 1
Trying RULE026/ORGANIZATION-2; RULE026 succeeded.
Conclude: TYPSYS of ORGANIZATION-2 is REAL-TIME (.8)
Conclude: OUTPUT of ORGANIZATION-2 is INDIVIDUAL-TERMINALS (.8)
Conclude: INSTALL of ORGANIZATION-2 is PYRAMIDAL (.5)
Conclude: TRG of ORGANIZATION-2 is UNAVAILABLE (.8)
Trying RULE025/ORGANIZATION-2;
   --[no rules to conclude TRAINING of ORGANIZATION-2]
   12) IS THE TASK LEADER'S LEVEL OF TECHNICAL TRAINING CONSIDERED TO
       BE HIGH, LOW, OR UNKNOWN?
   ** HIGH
RULE025 failed due to clause 1
Trying RULE024/ORGANIZATION-2; RULE024 succeeded.
Conclude: TYPSYS of ORGANIZATION-2 is REAL-TIME (.96)
Conclude: OUTPUT of ORGANIZATION-2 is INDIVIDUAL-TERMINALS (.96)
Conclude: INSTALL of ORGANIZATION-2 is PYRAMIDAL (.9)
Conclude: TRG of ORGANIZATION-2 is TRAIN-EXISTING-STAFF (.8)
RULE023 failed {in preview} due to clause 1
RULE022 failed {in preview} due to clause 1
RULE021 failed {in preview} due to clause 1
Trying RULE020/ORGANIZATION-2; RULE020 succeeded.
Conclude: TYPSYS of ORGANIZATION-2 is NON-REAL-TIME (.8)
Conclude: OUTPUT of ORGANIZATION-2 is LARGE-SCREEN-DISPLAYS (.8)
Conclude: INSTALL of ORGANIZATION-2 is DIVISIONAL (.8)
Conclude: TRG of ORGANIZATION-2 is TRAIN-EXISTING-STAFF (.96)
RULE019 failed {in preview} due to clause 1
RULE018 failed {in preview} due to clause 1
RULE017 failed {in preview} due to clause 1
RULE016 failed {in preview} due to clause 1
RULE015 failed {in preview} due to clause 1
RULE014 failed {in preview} due to clause 1
Trying RULE013/ORGANIZATION-2; RULE013 succeeded.
Conclude: TYPSYS of ORGANIZATION-2 is REAL-TIME (.992)
Conclude: OUTPUT of ORGANIZATION-2 is INDIVIDUAL-TERMINALS (.992)
Conclude: INSTALL of ORGANIZATION-2 is PYRAMIDAL (.98)
Conclude: TRG of ORGANIZATION-2 is DO-NOT-HIRE-SPECIALISTS (.96)
RULE009 failed {in preview} due to clause 1
Trying RULE008/ORGANIZATION-2; RULE008 succeeded.
Conclude: TYPSYS of ORGANIZATION-2 is REAL-TIME (.998)
Conclude: OUTPUT of ORGANIZATION-2 is INDIVIDUAL-TERMINALS (.998)
Conclude: INSTALL of ORGANIZATION-2 is PYRAMIDAL (.996)
Conclude: TRG of ORGANIZATION-2 is DO-NOT-HIRE-SPECIALISTS (.992)
RULE007 failed {in preview} due to clause 1
Trying RULE006/ORGANIZATION-2; RULE006 succeeded.
Conclude: TYPSYS of ORGANIZATION-2 is REAL-TIME (.999)
Conclude: OUTPUT of ORGANIZATION-2 is INDIVIDUAL-TERMINALS (.999)
Conclude: INSTALL of ORGANIZATION-2 is PYRAMIDAL (.999)
Conclude: TRG of ORGANIZATION-2 is DO-NOT-HIRE-SPECIALISTS (.998)
Trying RULE005/ORGANIZATION-2; RULE005 succeeded.
Conclude: TYPSYS of ORGANIZATION-2 is REAL-TIME (.999)
Conclude: OUTPUT of ORGANIZATION-2 is INDIVIDUAL-TERMINALS (.999)
Conclude: INSTALL of ORGANIZATION-2 is UNAVAILABLE (.75)
Conclude: TRG of ORGANIZATION-2 is HIRE-SPECIALISTS (.6)
Trying RULE004/ORGANIZATION-2; RULE004 succeeded.
Conclude: TYPSYS of ORGANIZATION-2 is REAL-TIME (.999)
```

```
Conclude: OUTPUT of ORGANIZATION-2 is INDIVIDUAL-TERMINALS (.999)
Conclude: INSTALL of ORGANIZATION-2 is PYRAMIDAL (.999)
Conclude: TRG of ORGANIZATION-2 is DO-NOT-HIRE-SPECIALISTS (.999)
RULE003 failed {in preview} due to clause 1
    --[2] Finished: TYPSYS of ORGANIZATION-2
    --[2] Findout: OUTPUT of ORGANIZATION-2
    --[2] Finished: OUTPUT of ORGANIZATION-2
    --[2] Findout: INSTALL of ORGANIZATION-2
    --[2] Finished: INSTALL of ORGANIZATION-2
    --[2] Findout: TRG of ORGANIZATION-2
    --[2] Finished: TRG of ORGANIZATION-2
RULE002 succeeded.
Conclude: DECAIDS of ORGANIZATION-2 is TEXT NIL
USE THE FOLLOWING TYPE SYSTEM REAL-TIME
THE OUTPUT SHOULD BE DISPLAYED ON INDIVIDUAL-TERMINALS
THE MANNER OF INSTALLATION SHOULD BE PYRAMIDAL
AND THE RECOMMENDATION FOR TRAINING AND ASSISTANCE IS
DO-NOT-HIRE-SPECIALISTS . (1.0)
    --[1] Finished: DECAIDS of ORGANIZATION-2
antecedent RULE001 succeeded.
Conclusions: THE DECISION are as follows:
          USE THE FOLLOWING TYPE SYSTEM REAL-TIME THE OUTPUT SHOULD BE
               DISPLAYED ON INDIVIDUAL-TERMINALS THE MANNER OF
               INSTALLATION SHOULD BE PYRAMIDAL AND THE RECOMMENDATION
               FOR TRAINING AND ASSISTANCE IS DO-NOT-HIRE-SPECIALISTS .

    --[1] Findout: UNKSTRUC of ORGANIZATION-2
Trying RULE033/ORGANIZATION-2; RULE033 succeeded.
Conclude: UNKSTRUC of ORGANIZATION-2 is CENTRALIZED (.8)
Trying RULE034/ORGANIZATION-2; RULE034 succeeded.
antecedent RULE035 succeeded.
          THE
               RECOMMENDED INFORMAL STRUCTURE TO USE IS CENTRALIZED.

Conclude: UNKSTRUC of ORGANIZATION-2 is TEXT NIL
THE
RECOMMENDED INFORMAL STRUCTURE TO USE IS CENTRALIZED (1.0)
    --[1] Finished: UNKSTRUC of ORGANIZATION-2
    --[1] Findout: FORSTRUC of ORGANIZATION-2
Trying RULE039/ORGANIZATION-2; RULE039 succeeded.
Conclude: FORSTRUC of ORGANIZATION-2 is LINE (.8)
Trying RULE040/ORGANIZATION-2; RULE040 succeeded.
antecedent RULE041 succeeded.
          THE RECOMMENDED FORMAL STRUCTURE TO USE IS LINE.

Conclude: FORSTRUC of ORGANIZATION-2 is TEXT NIL
THE RECOMMENDED FORMAL STRUCTURE TO USE IS LINE (1.0)
    --[1] Finished: FORSTRUC of ORGANIZATION-2
```

APPENDIX D

ADDITIONAL EMYCIN/DECAIDS PARAMETER PROPERTIES

Defining Contexts

MAINPROPS - a list of parameters to "trace" when a context of this type is created. Generally these are labdata parameters whose values will always be needed in a consultation (see PARAMETERS.DOC for a definition of "labdata"). The user will be asked for the value of each of these parameters as soon as a context is created. This often serves to present a more coherent dialog than would appear if each parameter were requested when it was first needed in a rule. It is also possible to have non-labdata parameters for mainprops if there is always something you want to deduce about a new context. The goal parameter(s) of a system will be found in the MAINPROPS list of the main (or root) context type; its placement here is what sets the consultation started.

PROPTYPE - an atom PROP-type which lists all parameters which pertain to this type of context, e.g., the PROPTYPE of PERSON in MYCIN is PROP-PT, which contains such parameters as NAME, AGE, SEX, etc. When applying a rule, the system uses this property to tell which context in the tree a particular parameter belongs to.

TYPE - an atom used to form the context identifier (by appending a numeral) for contexts of this type, e.g., in MYCIN the TYPE of POSCUL (positive culture) is CULTURE-.

RULETYPES - a list of all rule types applicable to this context (see RULES.DOC), e.g., RULETYPES of POSCUL is the list (CULRULES POSCULRULES).

SYN - a template used for translating contexts of this type in questions or rules. The SYN property is a list of entries (<parms> <form>), where <parms> is a list of one or more parameters of the context, and <form> is a simple list of words including the elements of <parms>. The parameters must appear in the same order in both <parms> and <forms>, and they must all be labdata. The atom * is used like a parameter in <parms> and <form> to represent the parent context.

The system scans the SYN property until it finds an entry for which it knows the values of all parameters in <parms> ; values which were not supplied by the user will not be used. Once an element of SYN has been selected, a translation will be constructed by replacing each parameter in <form> with that parameter's value; * will be replace by the translation (using the SYN property) of the parent context. <parms> and <form> may be punctuated with semicolons denoting "places to stop" if the translation is unambiguous so far. E.g., the element from the SYN property of an organism ((IDENT ; *) (the IDENT ; from *)) will result in "the Klebsiella" if there are no other organisms whose IDENT is Klebsiella, and "the Klebsiella from the blood culture" if there is.

The simplest SYN is of the form (((parm)(parm))), i.e., there is a parameter of the context whose value itself can stand

for the context, e.g., (((NAME) (NAME))). If there is no SYN prop, the context identifier (e.g., ORGANISM-1) will remain untranslated.

UNIQUE - For use in conjunction with the SYN prop: it controls whether the context identifier needs to appear in the translation, e.g. a typical non-unique phrase is 'the blood culture (CULTURE-2)', since it is possible to have more than one culture from the same site. If the UNIQUE property is T, the context identifier is omitted, e.g., the root context type should have its UNIQUE property T. If UNIQUE is the atom '?', this means to omit the identifier if the first try at translating the context is, in fact, unique; e.g., if, in the example above, CULTURE-2 were the only blood culture in the consult then its translation would be simple 'the blood culture'. Most context types are 'non-unique' and will not have a UNIQUE property. The property is not necessary even on unique context; it exists simply to reduce excess verbiage where possible.

If your system has no context tree, you need only fill in the properties listed above for the main (root) context type. If you have a non-trivial context tree, however, it is also necessary to supply the following properties for non-root types: PROMPT1ST (or PROMPTEVER), PROMPT2ND, ASSOCWITH, and OFFSPRING.

PROMPTEVER - the 'prompt' that will be printed when the first context of this type is created. Only context types that will ALWAYS be created have PROMPTEVERs; if you have to ask whether there are any contexts of this type, then there should be a PROMPT1ST instead. E.g., in MYCIN, there is always at least one KNOWNORG under every POSCUL (by definition a positive culture is one from which organisms grew), so KNOWNORG has the PROMPTEVER (The first organism isolated from * will be referred to as:). [In PROMPTEVER, PROMPT1ST, and PROMTP2ND, the * will be filled in by the parent context.]

PROMPT1ST - the prompt asking whether contexts of this type exist. Unlike PROMPTEVER, this is a real question and requires an answer. E.g., the PROMPT1ST of CURTHER is (Is * currently receiving therapy with any anitmicrobial agent?).

PROMPT2ND - the prompt asking whether additional contexts of this type exist (to be used after at least one context of this type has been created). Omission of this property indicates that there is never more than one instance of the context under the parent context. E.g., the PROMPT2ND of KNOWNORG is (Were any other organisms isolated from * ?).

ASSOCWITH - a list of ancestor context types, showing this context type's location in the context tree. E.g. the ASSOCWITH of KNOWNORG in MYCIN is (POSCUL PERSON) and the the ASSOCWITH of POSCUL is (PERSON). This means that a POSCUL context is directly below the patient context, and that a KNOWNORG context is directly below a POSCUL.

OFFSPRING - a list of descendent context types, indicating which types that can hang directly below a context of this type in the context tree.

Context types may have other optional properties. If the context type is ever to appear in a rule, it must have a TRANS for translation (see description of a TRANS property in PARAMETERS.DOC). The * in the TRANS property of a context type is filled in by a translation of the tree root (main context). If you plan to use the SUMMARY option, the context type will need a CNTXTSUMMARY property as described in FEATURES.SUMMARY. If parmaters of the context type are to be gathered in a block using the TAB option for tabular input, it will need the TABPARMS, TABHEAD, TERSEHEAD, TABSTOPS, and LEGALTERM properties as described in FEATURES.TABULAR-INPUT.

Properties of Parameters

Below is a list of the properties that a parameter can have. All parameters need a TRANS. If the value of the parameter is ever requested of the user, it needs a PROMPT and EXPECT, and if applicable, LABDATA. Numeric parameters should have a CHECK property, and possibly a DEFAULT.

TRANS - how to translate the parameter. The TRANS is list; if it contains the atom *, the latter will be filled with the translation of the context to which this parameter belongs (e.g. (the identity of *)). Special verbs, such as "is", "has" (all those on the list TRANSVERBS), as well as the word "not" should be present as lower-case literal atoms for correct translation of the negation when the parameter is used in rules.

PROMPT - how to ask for the parameter's value; no PROMPT means that it makes no sense to ask the user for the value. The PROMPT a list; when the question is asked, the * in the list is replaced by the translation of the context being asked about; for multivalued parameters which are not "ASKALL", the atom "(valu)" is replaced by the particular value being asked about.

EXPECT - the set of legal answers to questions asking about this parameter. A null EXPECT is implicitly (YES NO).

The most common form of the EXPECT property is a list of the values (atoms). If an element of the EXPECT list is itself a list rather than an atom, it will be a list of one element and that element is to be evaluated to produce a list of values. Usually the code to be evaluated will be the name of a list. This is useful when more than one parameter will have the same list of possible values. The code to be evaluated, however, may be arbitrary Lisp code which produces a list. This will be useful if the list of legal values depends on some previous answer; e.g., in MYCIN, valid answers for COLLECT (method of collecting a culture specimen) depends on SITE; the form evaluated may reference CNTXT - the object for which you are asking, and PARM - the parameter.

A few atomic EXPECTs are recognized:

ANY - no restriction of the value

NUMB - the value must be a number

POSNUMB - the value must be a positive number

DATE - the value is a date

LABDATA - To you find out the value of this parameter, first try asking the user. The original meaning was that the parameter was the result of a quantitative lab test; this has been generalized to be anything that the user is likely to know. If the user does not give a definite answer to the question, the system will use rules (if any exist) to deduce the value. For parameters that have no LABDATA property, the system first tries to conclude the value using rules, and only asks if no value was concluded (and a PROMPT exists).

DEFAULT - if a numeric-valued parameter, the default units. This allows the user to give the answer in a different unit, and the system will convert it to the default units. The rules assume that the value is given in the default units.

CHECK - A form to EVAL to make sure that the user's numeric response is 'reasonable'. The CKECK property has the form:

```
(CHECK VALU lower-bound upper-bound text confirm integer)
```

VALU will be bound to the user's numeric response to the question; you must supply the lower and upper bound. Text will be printed if VALU is not within the indicated range. Confirm can be T or NIL: if T, the user may confirm that the answer is correct; if NIL, an answer outside the range is always a mistake. Integer may be T or NIL; if T, the answer must be an integer.

MULTIVALUED - the parameter is multivalued. This means that it can have several different correct values at the same time. (E.g., ALLERGIC the patient may be allergic to more than one drug.) This is different form the normal case in which the parameter is assumed to have a single correct value, and different values that are concluded represent competing hypotheses as to the true value.

If the value of the MULTIVALUED property is T, a separate question will be asked for each value (e.g., 'Is the patient allergic to penicillin?'). If the value is the atom ASKALL, one question will be asked in which the user is expected to give all the values (e.g., 'Please list all the antibiotice to which the patient is allergic.').

The TRANS of a multivalued parameter is stated in the plural (e.g., (the drugs to which * is allergic)). This phrasing is necessary for proper translation throughout the system.

PROPERNOUN - if the value should be capitalized in translation (e.g., NAME), then the parameter should have PROPERNOUN property T.

LEGALVALS - Always the list of all legal values for this parm, but is omitted if redundant (which it is for most parameters). All multivalued parms have a LEGALVALS property. In addition those parameters with EXPECTs which are pieces of code also have one. When a parameter has no LEGALVALS, the legal values are assumed to be specified by the parameter's EXPECT property. A parameter with no EXPECT or LEGALVALS will be treated as a yes/no parameter; this affects its translation in many parts of the system. The LEGALVALS property may be of the same forms as an EXPECT property. Two atomic forms are recognized. The atom CNTXT indicates that this parameter takes other contexts as its value (e.g., in MYCIN, the TREATFOR property of the patient is the list of all the organisms in the context tree that should be treated). The atom TEXT means that the parameter takes arbitrary pieces or text as its value. This will probably be the case for some goal parameter - the text will be the system's final analysis or recommendatios.

CNTXTVAL - To be used when the LEGALVALS is the atom CNTXT. The value of this property is a list of PROP-VALs (context types) indicating that contexts of the specified type(s) can be values of this parameter. The value may also be a function of a context (?).

SPECIAL - indicates ambiguous answers to the PROMPT. Usually of form ((<ambiguous response> <request for clarification>) ...) May also be triples with 3rd element a default value in case user responds UNKNOWN.

XTRASPECIAL - indicates a response to a question that actually includes the values for more than one parameter; is usually a list of lists ((<response1> <parm1 value1> <parm2 value2> etc.) (<response2> etc.) meaning that the response given should be used to conclude the values for the parameters; may also be ((<response1> code) ...) meaning that the code should be executed when the response has been given; some entries are of the form (<code> <parm1 parm2>) meaning that if the code EVALs with the given response, then a new value will be indicated-- the new value is the value for parm1 and the user's response is the value for parm2 (e.g., yes/no parameters when an answer other than yes or no is given)

REPROMPT - more specific than original prompt, is printed out when the user enters '?' in response to the oiginal prompt.

APPENDIX E

EMYCIN/DECAIDS PREDICATE FUNCTIONS

Non-Numeric Predicates

In all the predicates that use VALU, VALU may be omitted for yes/no paramters. If present, VALU may be an atom, a simple list, or a list of value-cf pairs. The predicates SAME and THOUGHTNOT return numbers which will be minimized by $AND to determining the setting of TALLY. The other predicates return true (or 1000) or false (NIL).

DEFINITE[CNTXT,PARM]

Returns true if PARM of CNTXT is known with certainty (cf = 1.0; for yes/no parameters, also cf = -1.0).

Ex: (DEFINITE CNTXT IDENT)

The identity of the organism is known with certainty

DEFIS[CNTXT.PARM,VALU]

Returns true if PARM of CNTXT is known with certainty to be VALU (cf = 1.0).

Ex: (DEFIS CNTXT IDENT MYCOBACTERIUM-TB)

It is definite that the identity of the organism is Mycobacterium-tb

DEFNOT[CNTXT,PARM,VALU]

Returns true if PARM of CNTXT is definitely not VALU (cf = -1.0).

Ex: (DEFNOT CNTXT IDENT VIRUS)

It is definite that the identity of the organism is not Virus

KNOWN[CNTXT,PARM]

Returns true if the value of PARM of CNTXT is known (cf > .2; for yes/no parameters, also cf < -.2).

Ex: (KNOWN CNTXT IDENT)

The identity of the organism is known

MIGHTBE[CNTXT,PARM,VALU]

True if PARM of CNTXT might be VALU, i.e. there is no evidence against it (cf > -.2).

Ex: (MIGHTBE CNTXT ADEQUATE)

There is no evidence that the dose of the drug was not appropriate

NOTDEFINITE[CNTXT,PARM]

Returns true if PARM of CNTXT is not known with certainty (cf < 1.0; for yes/no parameters, -1.0 < cf < 1.0).

Ex: (NOTDEFINITE CNTXT GENUS)

The genus of the organism is not known with certainty

NOTDEFIS[CNTXT,PARM.VALU]

Returns true if PARM of CNTXT is thought to be VALU, but not with certainty (.2 < cf < 1.0).

Ex: (NOTDEFIS CNTXT IDENT CRYPTOCOCCUS)

It is suspected that the identity of the organism is cryptococcus

NOTDEFNOT[CNTXT,PARM,VALU]

Returns true if PARM of CNTXT is thought not to be VALU, but not with certainty (-1.0 < cf < -.2).

Ex: (NOTDEFNOT CNTXT IDENT E.COLI)
It is suspected that the identity of the organism is not E.coli

NOTKNOWN[CNTXT,PARM]

Returns true if PARM of CNTXT is not known (cf <= .2; for yes/no parameters, -.2 <= cf <= .2).

Ex: (NOTKNOWN CNTXT IDENT)
The identity of the organism is not known

NOTSAME[CNTXT,PARM,VALU]

The logical compliment of SAME, returns true if PARM of CNTXT is not thought to be VALU (cf <= .2).

Ex: (NOTSAME CNTXT SPECSTAIN)
Organisms were not seen on the stain of the culture

ONCEKNOWN[CNTXT,PARM,RETFLG]

Finds the value of PARM of CNTXT. If RETFLG is NIL, it means "you have found a value for PARM of CNTXT"; returns the same as KNOWN would. If RETFLG is T, it means "find out all you can about PARM of CNTXT": this causes tracing, but ONCEKNOWN will return true even if nothing was found out. Intended to be invoked last among the updated-by rules of PARM.

Ex: (ONCEKNOWN CNTXT SAMEBUG)
There is an organisms with possibly the same identity as this organism

ONCEKNOWN*[CNTXT,PARMS]

Traces PARMS of CNTXT and returns T regardless of the result of this tracing. This is like calling ONCEKNOWN repeatedly, once for each parameter in PARMS, with RETFLG set to T in the calls.

Ex: (ONCEKNOWN* CNTXT (QUOTE (CURTHER PRIORTHER)))
Information has been gathered about current drugs of the patient and prior drugs of the patient

SAME[CNTXT,PARM,VALU]

Checks to see if PARM of CNTXT is VALU returning the associated cf. Always returns a number; in a rule, $AND will consider the clause 'true' if this number greater than .2.

Ex: (SAME CNTXT SITE BLOOD)
The site of the culture is blood

THOUGHTNOT[CNTXT,PARM,VALU]

Checks to see if PARM of CNTXT is not VALU, i.e., there is evidence against it. Always returns a number; in a rule, $AND will consider the clause 'true' if this number greater than .2. The number that is returned is the negative of the cf associated with the triple (CNTXT PARM VALU), so the clause will be true if the cf associated with that triple is less than -.2. This is the algebraic negation of SAME, whereas NOTSAME is the logical negation.

Ex: (THOUGHTNOT CNTXT IDENT E.COLI)
There is evidence that the identity of the organism is not E.coli

VNOTKNOWN[CNTXT,PARM,VALU]
Returns true if it is not known whether the value of PARM of CNTXT is (or is not) VALU (-.2 <= cf <= .2).
Ex: (VNOTKNOWN CNTXT IDENT E.COLI)
It is not known whether the identity of the organism is E.coli

Numeric predicate functions

There are five numeric predicate functions to be used with parameters which take numbers or dates as their values. A parameter with DATE for its EXPECT property accepts a date as input, but internally stores the answer as the number of days ago (or since the time of the original consultation for stored cases). The Lisp functions PLUS, DIFFERENCE, MINUS, TIMES, FQUOTIENT, and EXPT have translations in case they are used within these numeric predicates. The translations of numeric expressions can be very wordy. To have an expression translated tersely (using arithmetic operator symbols instead of text), enclose the expression in a call to the function TRSEXP.

BETWEEN*[VALU,LLIM,ULIM]
True if LLIM <= VALU < ULIM
Ex: (BETWEEN* (VAL1 CNTXT AGE) 10 50)
The age of the patient is between 10 years and 50 years

GREATEQ*[X,Y]
True if X and Y are numbers and X >= Y.
Ex: (GREATEQ* (VAL1 CNTXT NUMPOS) 2)
The number of cultures from this site which were positive for this organism is greater than or equal to 2

GREATERP*[X,Y]
True if X and Y are numbers and X > Y.
Ex: (GREATERP* (VAL1 CNTXT CSFGLUC) 80)
The csf glucose value is greater than 80

LESSEQ*[X,Y]
True if X and Y are numbers and X <= Y.
Ex: (LESSEQ* (VAL1 CNTXT CSFGLUC) 80)
The csf glucose value is less than or equal to 80

LESSP*[X,Y]
True if X and Y are numbers and X < Y.
Ex: (LESSP* (VAL1 CNTXT CSFGLUC) 80)
The csf glucose value is less than to 80

Conclusion Functions

The functions in a rule's ACTION concludes about one or more context-parameter-value triple. A cf for the triple is specified in the rule's ACTION. This cf will be modified by the certainty of the rule's PREMISE. $AND sets TALLY to the certainty of the PREMISE, defined to be the minimum of the values (numbers) returned by evaluating the PREMISE clauses (only SAME and THOUGHTNOT return numbers).

If a triple already exists, this new cf is "combined" with the cf associated with that triple. Otherwise, the new cf itself is associated with the new triple.

CONCLUDE[CNTXT,PARM,VALUE,TALLY,NUM]

Concludes that PARM of CNTXT is VALUE. The conclusion made by this call will have a cf that is TALLY times NUM.

```
Ex:      (CONCLUDE CNTXT CONTAMINANT YES TALLY 400)
         There is weakly suggestive evidence (.4) that the organism is a
                contaminant
```

CONCLUDE*[CNTXT,PARM,TALLY;VALS]

Performs multiple CONCLUDE's for a single CNTXT and PARM. VALUS is a list of pairs (value cf); each value is concluded with the corresponding

cf.

```
Ex:      [CONCLUDE* CNTXT IDENT TALLY (QUOTE ((E.COLI 400)
                                             (KLEBSIELLA-PNEUMONIAE 300)
                                             (PROTEUS-MIRABILIS 300]
         There is evidence that the identity of the organism is e.coli (.4)
                klebsiella-pneumoniae (.3) proteus-mirabilis (.3)
```

CONCLUDET[CNTXT,SWITCHNUM,CASE,TALLY,PARM,VALUS]

Tabular rule concluding fn. Concludes that PARM of CNTXT is one or more of the values in VALUS, according to the value of SWITCHNUM.

SWITCHNUM is a form to evaluate which must return a number. It is generally a call to VAL1 for some numeric parameter.

VALUS is a list (VAL1 VAL2 ... VALn) of values for PARM.

CASE is a list of cases which test SWITCHNUM and supply cfs for the values in VALUS that is to be concluded. Possible cases currently are:

(LT NUM CF1 CF2 ... CFn) if SWITCHNUM < NUM, conclude that PARM is VALi with cf CFi;

(BT NUM1 NUM2 CF1 CF2 ... CFn) if NUM1 <= SWITCHNUM < NUM2, conclude that PARM is VALi with cf CFi;

(GE NUM CF1 CF2 ... CFn) if NUM <= SWITCHNUM, conclude that PARM is VALi with cf CFi;

(U CF1 CF2 ... CFn) if SWITCHNUM = NIL, conclude that PARM is VALi with cf CFi;

Each CFi must be included in each case; if a particular value doesn't apply, the corresponding cf can be 0.

```
Ex:      (CONCLUDET CNTXT (VAL1 CNTXT LENSIGN)

                (QUOTE ((BT 9 13 -400 -500)
                        (BT 13 20 -500 -400)
                        (GE 20 600 300)))
                TALLY TYPE (QUOTE (BACTERIAL VIRAL)))
```

The type of the infection is as follows:

If the duration of the neurological signs is:

a) between 9 days and 13 days then: not bacterial (.4), not viral (.5);

b) between 13 days and 20 days then: not bacterial (.5), not viral (.4);

c). greater or equal to 20 days then: bacterial (.6), viral (.3);

CONCLUDETEXT[CNTXT,PARM,VALUE,TALLY,NUM]

This function calls CONCLUDE for parameters whose values are arbitrary peices of text. It is a different function because it translates differently.

DO-ALL[X] NL*

For multiple conclusions - evaluates each of its arguments (rule conclusions).

Ex: (DO-ALL (CONCLUDE CNTXT IDENT LISTERIA TALLY 500)
(CONCLUDE CNTXT GENUS CORYNEBACTERIUM TALLY 500)
1) There is suggestive evidence (.5) that the identity of the organism is Listeria, and
2) There is suggestive evidence (.5) that the genus of the organism is Corynebacterium

DONTASK[CNTXT,PARM]

Conclusion function which says that PARM of CNTXT should not be asked (although it may be traced, if needed).

Ex: (DONTASK CNTXT CONFORM)
Don't ask about the growth conformation of the organism

NOTRELEVANT[CNTXT,PARMS,CF]

A conclusion function that indicates that the value of each of the parameters in PARMs is not relevant for CNTXT: we shouldn't ever ask or try rules to deduce the value.

Ex: (NOTRELEVANT CNTXT (QUOTE (SECONDARY)) 1000)

It is definite (1.0) that the following is irrelevant: the infection to which the bacteremia is secondary

PRINTCONCLUSIONS[CNTXT,PARM,HEADER]

Displays nicely the value of PARM of CNTXT, or indicates that no conclusions were made. Intended as a simple goal rule ACTION. If HEADER is T, prefixes values with a simple header announcing what these are the values of; if other non-NIL value, HEADER is printed; otherwise no header at all is printed, and nothing is mentioned if there are no values.

Special facility for use with TEXT-valued parms: if PARM has a property LABEL.ORDER, then TEXT values of equal cf will be sorted by their labels. For LABEL.ORDER=T, the labels are integers, and the values will be sorted in ascending order of label; otherwise the LABEL.ORDER property is a list of labels (atoms), and the values are sorted according to the order of the labels in this list.

Ex: (PRINTCONCLUSIONS CNTXT REGIMEN)
Display the therapeutic regimen of the patient

Auxilliary functions

$AND[$$CLAUSES] NL*

Evaluates each of the predicates in $$CLAUSES until one fails; if all succeed (i.e. return T or a cf > .2) the minimum cf is returned, else NIL. All rule premises and the predicates of all mapping functions must be calls to $AND - even if there is only a single predicate clause inside the $AND.

$OR[$$CLAUSES] NL*

Evaluates each of the predicates in $$CLAUSES, returning the c.f. of the most highly confirmed clause, unless all fail. Stops if one of the clauses evaluates with certainty (since the maximum 1.0 is the result).

```
Ex:         ($OR (GREATERP* (VAL1 CNTXT WBC) 12.5)
                 (GREATERP* (VAL1 CNTXT PMNS) 80)
                 (GREATERP* (VAL1 CNTXT BANDS) 10))
```

1) The white count from the patient's peripheral CBC (in thousands) is greater than 12.5, or
2) The percent of PMN's in the CBC is greater than 80, or
3) The percent of peripheral WBC's which are immature in the CBC is greater than 10

LISTOF[N] L*

Simply EVALs its argument (which is usually the name of a list). Used as argument to a basic predicate when a choice of values is indicated.

```
Ex:      (SAME CNTXT SITE (LISTOF STERILESITES))
```

The site of the culture is one of: those sites that are normally sterile

ONEOF[X] NL*

A no-spread quote: returns its argument list. Used as argument to a basic predicate when a choice of values is indicated.

```
Ex:      (SAME CNTXT SITE (ONEOF URINE SPUTUM))
```

The site of the culture is one of: urine sputum

QUOTE[X] NL*

Lisp function. It is used in action functions that require a list of parameters, values, etc. (e.g., CONCLUDET).

TEXT[N] L*

Constructs value of TEXT-valued parms. The first argument is a label (or NIL) which may be used to tag the value for sorting by PRINTCONCLUSIONS. The remaining args are arbitrary rule forms to construct a text phrase. Result is a list (TEXT label . phrase). If there is only one arg, the label is interpreted as a TEXT tag, i.e. (TEXT label) is the same as (TEXT label (TEXTAG label)).

TEXTAG[TAG] NL

Quotes a text 'tag', a place holder for a string of text which is the 'value' of a conclusion parameter. TAG should be in PROP-TEXT, and should have a TRANS which is the string in question (at least in current implementation).

UNITS[U] NL*

Returns its first argument. The second argument is not seen by the function, but is a unit and is used for translation.

```
Ex:      (GREATERP* (VAL1 CNTXT AGE) (UNITS 3 YEARS))
```

The age of the patient is greater than three years

VAL[CNTXT,PARM]

Returns PARM of CNTXT as a list of pairs (value cf), tracing the parameter first if it has not been traced yet.

VAL1[ATM,PARM] NL

Returns the value of PARM of ATM, without its cf. Only suitable for single-valued parameters. ATM is evaluated, PARM is not.

VALYEW[CNTXT,PARM]

Returns PARM of CNTXT as a list of pairs (value cf) in order of decreasing cf. This is the same as returned by VAL if the parameter has already been traced. VALYEW causes no tracing; it can be interpreted as the system's current information about PARM of CNTXT

Mapping functions

You probably won't need to use mapping functions in any of your rules. The exist to allow a rule to use parameters of the context to which the rule is applied, as well as parameters of each of a specified list of contexts.

The functions map over a list $$MAPSET which is usually a list of contexts. $$FREEVAR is the name of the iteration variable; it is not necessary to specify a value for $$FREEVAR, the default FREEVAR will be used if none is specified. $$PRED is a predicate which has the same form as a rule PREMISE. Clauses in $$PRED may use CNTXT (the context to which the rule is being applied) as well as FREEVAR in their context slot.
If $$MAPSET is a list of pairs (which it will be if the set is the result of a call to GETALL or GETOFFSPRING), the CARS flag should be set to T. This indicates that each time $$FREEVAR should be set to the CAR of the current element rather than the element itself. Most of the functions return $$ANSET set to their result. For functions whose result is a list, COLLECTEDLST is the default for $$ANSET; for those that return a single element, the default is FOUNDVAR. These result variables are global; the result of a mapping function in a rule's PREMISE is often used in that rule's ACTION.

A mapping function can be embedding in a call to the LISP function NOT, and the action and translations will be appropriatly negated.

FINDMAX[$$MAPSET,$$PRED,$$TEST,$$FREEVAR,$$ANSET,CARS] NL

Mapping function that returns $$ANSET set to the element of $$MAPSET which had the largest value of $$TEST out of all those elements which satisfied $$PRED. Global MAXVAL is set to this maximum value of $$TEST.

```
Ex:      (FINDMAX (GETALL CURTHER)
                  ($AND (KNOWN $FREEDRUG WHENSTART))
                  (VAL1 $FREEDRUG WHENSTART)
                  $FREEDRUG NIL T)
```

You have examined the current drugs of the patient for which the time since therapy with this drug was started is known, and have selected the one having the maximum value for the time since therapy with this drug was started

FINDMIN[$$MAPSET,$$PRED,$$TEST,$$FREEVAR,$$ANSET,CARS] NL

Like FINDMAX, but looks for the the smallest value of $$TEST, and sets global MINVAL.

Ex:
```
        (FINDMIN (GETALL PRIORTHER)
                 ($AND (LESSEQ* (VAL1 $FREEDRUG WHENSTOP) 5))
                 (VAL1 $FREEDRUG WHENSTOP)
                 $FREEDRUG NIL T
```

You have examined the prior drugs of the patient for which the time since therapy with this drug was discontinued is less than or equal to 5 days, and have selected the one having the minimum value for the time since therapy with this drug was discontinued

FORALL[$$MAPSET,$$PRED,$$FREEVAR,CARS] NL

True if $$PRED is true for each element of $$MAPSET. Function returns (trivially) true if map set is empty.

Ex:
```
        (FORALL (GETALL POSCUL) ($AND (NOTSAME FREEVAR SPECSTAIN)
                                      (NOTSAME FREEVAR CRYPTO-SEROLOGY)
                                      (NOTSAME FREEVAR COCCI-SEROLOGY)))
```

For each of the the positive cultures of the patient it is true that

1) Organisms were not seen on the stain of this culture,
2) The cryptococcal antigen in the csf was not positive, and
3) The csf coccidioides serology was not positive

THEREARE[$$MAPSET,$$PRED,$$FREEVAR,$$ANSET,CARS,DUPLES] NL

Collects all the elements of $$MAPSET for which $$PRED is true. If DUPLES is T, it returns a list of duples pairing each element of $$MAPSET that succeeded with the value (number) returned when the predicate was appliedto that element.

Ex:
```
        (THEREARE (GETALL KNOWNORG)
                  ($AND (DEFINITE CNTXT IDENT))
                  NIL COLLECTEDORGS T)
```

You have examined the organisms isolated from positive cultures obtained from the patient, selecting those for which the identity of the organism is known with certainty

THEREARE![N] L*

Returns true if LST is non-empty. This is like a call to THEREARE with $$MAPSET T, but translates better.

Ex:
```
        (THEREARE! (GETOFFSPRING CNTXT SMEARORG))
```

There are organisms noted on smears of this culture

THEREXISTS[$$MAPSET,$$PRED,$$FREEVAR,$$ANSET,CARS] NL

Like THEREARE, but just finds the FIRST element (or CAR of element) satisfying $$PRED, and returns that.

Ex:
```
        (THEREXISTS (GETALL CURTHER)
                    ($AND (SAME FREEVAR DNAME (ONEOF AMPICILLIN CARBENICILLIN
                                                     PENICILLIN METHICILLIN)))
                    NIL NIL T)
```

You have examined current drugs of the patient, and have found one for which the name of this drug is one of: ampicillin carbenicillin penicillin methicillin

Functions used within Mapping Functions

GETALL[CTYPE] NL

Returns a list of all contexts of type CTYPE. Currently in the form ((cntxt 1000)...), until we get around to being neater. This is often used in the $$MAPSET slot.

GETOFFSPRING[CNTXT,TYPE]

Returns a list of contexts of type TYPE descendant to CNTXT. Currently a list ((cntxt 1000) ...).

APPEND[L] L*

Lisp function. It is used in the $$MAPSET slot when more than one type of context is to be examined.

NOTSAMEANS[CNTXT1,CNTXT2,PARM]

Premise clause which is true if CNTXT1 and CNTXT2 have different values

SAMEANS[CNTXT1,CNTXT2,PARM]

A premise function that is true if CNTXT1 and CNTXT2 have the same value for parameter PARM.

TRACEDP[CNTXT,PARM]

True if PARM has been traced for CNTXT. This is used when the value of a parameter of one context is to be transfered to another context. To avoid circular reasoning, we specify that the target parameter must already be traced for a context to satisfy the predicate.

```
Ex:      (THEREXISTS (APPEND (GETALL POSCUL) (GETALL PENDCUL))
                   ($AND   (TRACEDP FREEVAR NOSOCOMIAL)
                           (KNOWN FREEVAR NOSOCOMIAL)
                           (SAMEANS CNTXT FREEVAR SITE))
                   NIL FOUNDCUL T)
```

You have examined positive cultures obtained from the patient and pending cultures of the patient, and have found one for which

1) All information about whether the infection was acquired while the patient was hospitalized has been gathered, and
2) It is known whether the infection was acquired while the patient was hospitalized, and
3) The culture under consideration and this culture have the same value for the site of the culture

Action Functions used in Rules with Mapping Functions

CONCLIST[CNTXT,PARM,GVAL,TALLY]

GVAL is a list of duples (value cf). Concludes that PARM of CNTXT is each of those values, modified by TALLY.

```
Ex:      (CONCLIST CNTXT IDENT GRIDVAL 900)
```

There is strongly suggestive evidence (.9) that each of the ones that you found is the identity of the organism

CONCLUDEALL[CNTXTS,PARM,VALU,CF]

Makes the same conclusion for each of a list of contexts.

```
Ex:      (CONCLUDEALL COLLECTEDCULS REGTHER YES -1000)
```

It is definite (1.0) that the organisms isolated from the cultures that you selected should not be considered for therapy

TRANSDIFPARM[FROM!,FPARM,TO!,TPARM,CF,POSITIVE]

Transfers the value of FPARM of FROM! to TPARM of TO!, modified by CF. Either FROM! or TO! may be a list of contexts or a single context. FPARM and TPARM are different parameters. If POSITIVE is set, it only transfers values with non-negative CFs.

Ex: (TRANSDIFPARM COLLECTEDORGS IDENT CNTXT COVERFOR 700)

There is suggestive evidence (.7) that the identity of each of the organisms that you selected is the organisms (other than those seen on cultures or smears) which might be causing the infection

TRANSLIST[FROM,TO,PARMS,I]

Transfers to context TO the values of of each of the PARMS of the contexts in FROM, modifying the cf's by I.

Ex: (TRANSLIST (VALYEW CNTXT SAMEBUG) CNTXT (QUOTE (IDENT)) 1000)

It is definite (1.0) that these properties - ident - should be transferred from the organisms with possibly the same identity as this organism to this organism

TRANSPARM[FROM!,TO!,PARM,CF]

Transfers the value of PARM of FROM! to TO!, modified by CF. Either FROM! or TO! may be a list of contexts or a single context. If POSITIVE is set, only transfers values with non-negative CFs.

Ex: (TRANSPARM FOUNDCUL CNTXT SECONDARY 1000)

It is definite (1.0) that the information that you have gathered about the infection to which the bacteremia is secondary is also relevant to this culture

APPENDIX F

DECAIDS USER'S PROCEDURES

In order to present the procedures necessary to use the DECAIDS prototype decision support program, the following quick-reference material is provided. All TOPS-20 and EMYCIN/ DECAIDS commands must be followed by a carriage return to enter the user command or response. A sample consultation is presented in Appendix C.

1. Select the desired ARPANET compatible terminal to be used.

2. Connect to the local ARPANET TIP.

3. If using dial-up device, upon receiving the carrier tone, connect the telephone to the terminal's modem.

4. Depress the "RETURN" key once.

5. The following will be printed by the system:

```
NPS TIP 420#:  2
```

6. The user next enters

```
@1 116
```

This will connect the user to the University of Southern California's Information Sciences Institute System E computer referred to as ISIE.

7. The system will respond with:

```
TRYING ...
OPEN
ISI-SYSTEM-E.  TOPS-20 MONITOR 3A(105)
@
```

The "@" symbol is the TOPS-20 operating system's prompt. After this symbol, a user may enter his commands or responses.

8. The user next enters:

```
LOG DECAIDS escape
```

9. On the same line with the last entry, the user will be challenged with:

```
(PASSWORD)
```

10. The user must respond to (PASSWORD) with:

```
"PASSWORD" escape.
```

(The actual password may be obtained from LTCOL R. J. Roland or Professor Gary Poock.)

11. Again on the same line as above, the user will be challenged with:

```
(ACCOUNT)
```

12. The user's proper response to (ACCOUNT) is to enter a carriage return.

13. The current TOPS-20 operating system will respond with accounting data, date, and user file information. This information may be viewed or terminated with:

```
Control O
```

14. At the end of the login information the "@" prompt will be returned by the TOPS-20 operating system. At any time after logging in has been completed, the user may enter a:

```
Control C
```

in order to return to an "@" prompt and thereby facilitate a quick log-off with the command:

```
LOGO
```

15. To enter the EMYCIN/DECAIDS system, the user must type:

```
EMYCIN.EXE
```

16. The EMYCIN.EXE will respond with:

```
LOADING CHANGES ...
FILE CREATED (date) and (time)
CHANGESCOMS
(<DECAIDS> CHANGES..current number)
(<DECAIDS> EMYCIN.EXE.8.<DECAIDS>LISP.EXE.80516)

_
```

17. The "_" is the EMYCIN prompt symbol after which DECAIDS commands may be issued.

18. To begin the DECAIDS consultation program, the user enters:

```
BEGIN]
```

The right square bracket must be entered. A short delay (10 to 30 seconds) may be experienced before the next system response occurs.

19. The user will next be challenged with:

```
SPECIAL OPTIONS (TYPE ? for HELP)
```

20. The user may respond with a carriage return if no special options, such as fault tracing (i.e., FT 1 or 2 or 3 or 4), are desired or with a "?" if an explanation of available options is desired.

21. The system will next challenge the user with:

```
INSTRUCTIONS (Y or N)
```

22. On the same line a "Y" response to 21 above will present a line of instructions on the use of the DECAIDS

system and an "N" will continue with the DECAIDS consultation session.

23. DECAIDS will continue with the consultation by presenting the user with:

```
(current date) and (current time)

______________ORGANIZATION=1___________

1)  THIS PROGRAM IS DESIGNED TO PROVIDE
    MANAGERS AT ALL LEVELS WITH ADVICE
    CONCERNING THE USE OF THEIR COMPUTER
    RESOURCES.  IN ORDER TO PROVIDE THIS
    INFORMATION, THE USER WILL BE ASKED TO
    FURNISH DATA CONCERNING:  HIS ORGANI-
    ZATION, ITS LEVEL OF TRAINING, THE
    ORGANIZATION"S LEADER, THE ENVIRON-
    MENT AFFECTING THE DECISION, AND THE
    TASK FACING THE ORGANIZATION.  WHAT
    IS THE TYPE OF PROBLEM WHICH THE
    ORGANIZATION FACES?
```

This is the beginning of the consultation session. This first question about the type of problem may be answered with any subject name, for example: electronic warfare. A full consultation session is presented in Appendix C.

24. The consultation will continue by asking ten additional questions for the user to answer at the end of which the user will be presented with the DECAIDS recommendations.

25. After the recommendations are offered, the system will ask:

Do you wish advice on another patient?

(Based originally on a medical background, the inference engine continues to ask for "patients.")

26. A user response of "Y" will start another consultation session with a title of ORGANIZATION-2 and an "N" response will return the user to the TOPS-20 operation system and its prompt of:

```
@
```

27. To log off the system from the TOPS-20 operating system the user need only enter:

LOGO carriage return

28. The system will respond with:

```
KILLED JOB (#), USER DECAIDS, ACCOUNT NPS-OTHER-
STUDENTS, TTY 167, AT (date, USED (time)
CLOSED
```

and the session is closed.

APPENDIX G

DECAIDS KNOWLEDGE BASE MODIFICATION PROCEDURES

The following procedures are provided as a quick reference to fill in a new knowledge base or modify an existing one. The system designer will be required to declare parameters, define rules, and to save (make a file called CHANGES) the declarations and definitions. The parameters declared may be context names, rule group names, or value parameter names. The EMCYIN system prompts (requests for values from the designer) for the types of parameters mentioned above will be summarized in this section and examples will be provided. A carriage return is to be typed after each sample command. Entering the EMYCIN file is accomplished as described in Appendix F.

A. DECLARING PARAMETERS

1. To declare parameters the designer enters the following command after the EMYCIN prompt of "_"

```
GETPARMS
```

2. The system will respond with:

```
PARAMETER NAME:
```

3. If the system designer then responds with the name of a new parameter, the system will commence prompting for parameter property values.

4. The first prompt for a property value will be:

```
PROPGROUP:
```

5. A response to PROPGROUP of
   a. ALLNAMES: signals the system that the parameter is to be a rulegroup name
   b. PROP-VAL signals that the parameter is to be a context name, and
   c. PROP-(name) signals that the parameter is to be a value parameter in the parameter grouping of (name) which is of the designer's choosing.
6. RULE GROUP DECLARATIONS
   a. If the parameter is to be a rule group, then the next property prompt will be

      CONTEXT?

   b. The designer should respond with the context(s) names to which the rules of the named rule group will apply, i.e.:

      ORGANIZATION

   c. The next property prompt will be:

      SVAL:

   d. The designer shall respond with the appropriate context name, i.e.:

      ORGANIZATION

   e. The following prompt will be for:

      CTRANS:

   f. The appropriate context name is the correct designer response, i.e.:

      ORGANIZATION

g. The next prompt will be for:

    PROPTYPE:

h. Here, the designer should respond with:

    PROP-ALLNAMES

   The system will return with a prompt for:

    SUBPROPERTY:

   A carriage return after "SUBPROPERTY" will return a system response of:

    PARAMETER NAME:

   A carriage return after "PARAMETER NAME" will take the user out of the GETPARMS subprogram and return a:

    DONE

7. CONTEXT DECLARATIONS

   a. If the response to PROPGROUP is PROP-VAL, then the first prompt for a context parameter will be:

       TRANS:

   b. The designer response to "TRANS:" is the designer's literal interpretation of his intent for this context name, i.e.:

       (the organization)

   c. The next property prompt is:

       MAINPROPS:

   d. The designer response may be a carriage return if no MAINPROPS are to be used or a list of parameter names, i.e.:

       (PROBTYPE TASK STRUCTURE TECHNOLOGY)

e. The context parameter's next property prompt will be:

    PROPTYPE:

f. The designer should respond to PROPTYPE with the value parameter group(s) names to which this context will apply, i.e.:

    PROP-ORG

g. The next property prompt will be:

    TYPE:

h. The correct designer response to "TYPE:" is the appropriate context name, i.e.:

    ORGANIZATION

i. The system will next prompt for a response to:

    RULETYPES:

j. Here, the proper response is the rule group(s) names to which the context will apply, i.e.:

    (ORGRULES)

k. The final property prompt seen in context declarations in DECAIDS is:

    GOALS:

l. The designer should respond with those goal-parameters for the current context, i.e.:

    (DECAIDS FORMAL UNKSTRUC)

m. After prompting for the standard property values listed above, the system will request:

    SUBPROPERTY:

n. If the designer has a need to use additional property values, such as LABDATA, then he should respond to "SUBPROPERTY:" with the name of that property which he should use, i.e.:

    LABDATA

o. The system will then prompt the designer to provide a value for the subproperty just defined, i.e.:

    LABDATA:

p. A proper response to "LABDATA:" is:

    T

q. When the designer has completed declaring parameters, a carriage return should be entered to the systems request for another subproperty definition.

r. The system will next return with:

    PARAMETER NAME:

   A carriage return response here will cause a system response of:

    DONE

   and return the designer to the EMYCIN.EXE file with its "_" prompt.

8. SAVING FILES

   a. The above work is saved in a CHANGES file with the following command:

    MF CHANGES

   b. The EMYCIN file will return the now current edition number of the CHANGES file.

9. CHANGING A PROPERTY VALUE

   a. Changes to a property value are made by typing the property name while still in the GETPARMS subprogram, i.e.:

```
PARAMETER NAME:  STRESS
SUBPROPERTY:  TRANS
```

   b. The system will return that subproperty to the designer expecting a new value to be entered, i.e.:

```
TRANS:
```

   c. The designer should enter a new value and a carriage return, i.e.:

```
(THE NEW ORGANIZATION)   carriage return
```

   d. The system will challenge with

```
[NEW VALUE]
```

   e. On the same line as "[NEW VALUE]," the designer must respond with:

```
YES, for "YES" or N, for "NO"
```

   f. The system will then continue prompting with:

```
SUBPROPERTY:
```

B. DEFINING RULES

1. Rules are defined by initiating a call to the GETRULES subprogram with:

```
GETRULES]
```

2. The system will respond with:

```
RULE#, NEW or SUBJECT FOR NEW RULE:
```

3. To enter a new rule, the designer must respond with:

```
NEW
```

To edit an old rule, the designer must respond with the desired rule number to be edited.

4. After "NEW" is typed by the designer, the system will respond with:

```
ANTECEDENT RULE?
```

5. In most cases the rule will not be an antecedent rule and the correct response is simply:

```
N
```

6. The system next sends:

```
RULE (number)
PREMISE:
```

7. For a new rule, the designer should first define a rule premise and should respond to 6 above with his premise statement, in the INTERLISP syntax, i.e."

```
($AND (SAME CNTXT STRESS LOW))
```

8. The systems response will be either:

   a. an error message for syntax or undeclared parameter,

   or

   b. ```
      RULE (number)
      ACTION:
      ```

9. In response to the system's request for the ACTION statement, the designer should enter the rule's appropriate action statement, i.e.:
      ```

```
(CONCLUDE CNTXT SIZE LARGE TALLY 900)
```

The "TALLY 900" is the designer's certainty factor entry.

10. The system may again respond with an error message or return:

```
SUBJECT OF RULE (number) IS (rule group name)
[CONFIRM[
```

11. If (rule group) is the correct name to which the rule belongs, then enter

```
Y
```

immediately after [CONFIRM], i.e.:

```
[CONFIRM] Y
```

12. If in response to "RULE#, NEW or SUBJECT FOR NEW RULE:," the designer enters a rule number, the system will return:

```
TRANSLATE, DELETE, NO CHANGE, OR NAME
OF PROP TO MODIFY:
```

13. The designer may now specify the premise or action statement if he chooses to edit either, i.e.:

```
TRANSLATE, DELETE, NO CHANGE, OR NAME OR
PROP TO MODIFY:  PREMISE
```

14. The system will respond with the current premise value and on the next line return:

```
PREMISE:
```

awaiting the new value.

15. After the new value has been entered, the system will challenge with:

```
[NEW VALUE]:
```

a "Y" for "yes" or "N" for "no" immediately after [NEW VALUE] is the correct response.

16. A carriage return after RULE#, NEW or...will cause the system to return:

```
DONE
```

17. Rules may now be saved with:

```
MF CHANGES
```

18. Debugging of error messages may be facilitated by reference to the XEROX INTERLISP Manual and via communications with the AI personnel at Stanford University. (ARPANET address:

SCOTT@@@SUMEX-AIM

Carlisle Scott is a programmer with the AI group at Stanford University who has provided a great deal of assistance in learning the EMYCIN system.)

C. PRINTING PARAMETERS AND RULES

1. The parameters list is printed out with the following command:

```
PRINTPARMS (NIL T 72 T)
```

2. The rules may be listed with the following command:

```
(PRINTRULES rulegroup name 'B)
```

D. LOGGING OFF FROM THE DECAIDS FULE

1. From the DECAIDS file a CONTROL-C command and no carriage return will return the system designer or consultation user to the TOPS-20 operating system.

2. The command:

```
LOGO
```

will log the system user off of the computer at ISIE and off of the ARPANET TIP.

3. The system will terminate with:

```
KILLED JOB #, USER DECAIDS, ACCOUNT NPF-
OTHER-STUDENTS, TTY 135, at (date time),
USED (time) CLOSED.
```

BIBLIOGRAPHY

Davis, R., Buchanan, B., and Shortliffe, E., "Production Rules as a Representation for a Knowledge-Based Consultation Program," Artificial Intelligence, v. 8, no. 1, p. 15-43, February 1977.

Nilsson, N. J., Problem Solving Methods in Artificial Intelligence, p. 87-115, McGraw-Hill, 1971.

Nilsson, N. J., Artificial Intelligence -- A Framework, p. 1-45, Draft of 24 May 1978, by permission.

Office of Naval Research Final Technical Report, The Impact of Computer-Based Decision Aids on Organization Structure in the Task Force Staff, by B. I. Spector, R. E. Hayes, and M. J. Crain, September 1976.

Sacerdoti, E., "ARPANET Netnotes Communications concerning Artificial Intelligence," May 1979.

Scott, C., "ARPANET Netnotes Communications concerning Artificial Intelligence and the EMYCIN Program," January-June 1979.

Stanford Research Institute, Artificial Intelligence and National Security -- Some Opportunities, by P. E. Hart, p. 2-5, March 1978.

Teitelman, W., and others, INTERLISP Manual, Xerox Corporation, 1974.

Tosi, H. L. and Carroll, S. J., Management: Contingencies, Structures, and Processes, St. Clair Press, 1976.

Waterman, D. A., An Introduction to Production Systems, paper issued by the Rand Corporation, November 1976.

Williams, L. R., Modern Approaches to Defense Decision Making, USAWC Military Studies Program Paper, Carlisle Barracks, Pennsylvania, 12 May 1978.

Winston, P. H., Artificial Intelligence, p. 102, 136, 144, 148, 243-245, 273-275, 323-324, 357, 433, Addison-Wesley, 1977.

INITIAL DISTRIBUTION LIST

No. Copies

1. Defense Documentation Center 2
Cameron Station
Alexandria, Virginia 22314

2. Library, Code 0142 2
Naval Postgraduate School
Monterey, California 93940

3. Department Chairman, Code 52 1
Department of Computer Science
Naval Postgraduate School
Monterey, California 93940

4. LTCOL R. J. Roland, USAF, Code 52 Rl 4
Department of Computer Science
Naval Postgraduate School
Monterey, California 93940

5. Professor G. K. Poock, Code 55 Pk 4
Department of Operations Research
Naval Postgraduate School
Monterey, California 93940

6. MAJ Thomas Buscemi, Jr., USMC 2
Programming Support Division
Marine Corps Tactical Systems Support Activity
Marine Corps Base, Camp Pendleton
Oceanside, California 96055

7. LCDR J. M. Masica, USN 2
Helicopter Combat Support Squadron Six
Naval Air Station
Norfolk, Virginia 23511

8. Dr. P. J. Buscemi 1
Department of Material Sciences
and Engineering
University of Florida
Gainesville, Florida 32611

9. Dr. E. Sacerdoti 1
Stanford Research Institute
Artificial Intelligence Center
333 Ravenswood Avenue
Menlo Park, California 94025

10. Dr. R. H. Anderson 1
The Rand Corporation
1700 Main Street
Santa Monica, California 90406

11. Dr. G. A. Rahe Code 52Ra 1
Department of Computer Science
Naval Postgraduate School
Monterey, California 93940

12. Professor J. M. Wozencraft, Code 74 1
Command, Control and Communications Group
Naval Postgraduate School
Monterey, California 93940

13. Dr. M. Denicoff, Code 437 1
Office of Naval Research
800 North Quincy Street
Arlington, Virginia 22217

14. Mr. C. C. Stout 1
Naval Electronics Systems Command
Code 330
2511 Jefferson Davis Highway
Arlington, Virginia 20360

15. Dr. M. Tolcott, Code 455 1
Office of Naval Research
800 North Quincy Street
Arlington, Virginia 22217

183364

Thesis
B888 Buscemi
c.1 A prototype production rule program for a decision support system.

thesB888
A prototype production rule program for
3 2768 002 08816 3
DUDLEY KNOX LIBRARY